

[MS-EMF]: Enhanced Metafile Format

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
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.
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.
03/14/2008	4.0	Major	Abstract data model and Windows version-specific behavior added.
05/16/2008	4.0.1	Editorial	Revised and edited the technical content.
06/20/2008	4.1	Minor	Updated the technical content.
07/25/2008	4.1.1	Editorial	Revised and edited the technical content.
08/29/2008	4.2	Minor	Updated the technical content.
10/24/2008	5.0	Major	Updated and revised the technical content.
12/05/2008	5.1	Minor	Updated the technical content.
01/16/2009	6.0	Major	Updated and revised the technical content.
02/27/2009	6.1	Minor	Updated the technical content.
04/10/2009	6.2	Minor	Updated the technical content.
05/22/2009	6.2.1	Editorial	Revised and edited the technical content.
07/02/2009	6.3	Minor	Updated the technical content.
08/14/2009	6.4	Minor	Updated the technical content.
09/25/2009	6.5	Minor	Updated the technical content.
11/06/2009	6.5.1	Editorial	Revised and edited the technical content.
12/18/2009	6.6	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
01/29/2010	6.6.1	Editorial	Revised and edited the technical content.
03/12/2010	6.7	Minor	Updated the technical content.
04/23/2010	6.7.1	Editorial	Revised and edited the technical content.
06/04/2010	6.8	Minor	Updated the technical content.
07/16/2010	6.8	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	6.8	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	6.9	Minor	Clarified the meaning of the technical content.
11/19/2010	7.0	Major	Significantly changed the technical content.
01/07/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	7.1	Minor	Clarified the meaning of the technical content.
09/23/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	8.0	Major	Significantly changed the technical content.
03/30/2012	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	8.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	9.0	Major	Significantly changed the technical content.
11/14/2013	9.0	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
02/13/2014	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/15/2014	10.0	Major	Significantly changed the technical content.

Contents

1	Introduction	10
1.1	Glossary	10
1.2	References	17
1.2.1	Normative References	17
1.2.2	Informative References	17
1.3	Overview	17
1.3.1	Metafile Structure	17
1.3.2	Graphics Objects	19
1.3.3	Byte Ordering	20
1.4	Relationship to Protocols and Other Structures	20
1.5	Applicability Statement	20
1.6	Versioning and Localization	20
1.7	Vendor-Extensible Fields	21
2	Structures	22
2.1	EMF Enumerations	22
2.1.1	RecordType Enumeration	22
2.1.2	ArcDirection Enumeration	30
2.1.3	ArmStyle Enumeration	31
2.1.4	BackgroundMode Enumeration	32
2.1.5	ColorAdjustment Enumeration	32
2.1.6	ColorMatchToTarget Enumeration	32
2.1.7	ColorSpace Enumeration	33
2.1.8	Contrast Enumeration	33
2.1.9	DIBColors Enumeration	34
2.1.10	EmrComment Enumeration	34
2.1.11	ExtTextOutOptions Enumeration	35
2.1.12	FamilyType Enumeration	36
2.1.13	FloodFill Enumeration	36
2.1.14	FormatSignature Enumeration	36
2.1.15	GradientFill Enumeration	37
2.1.16	GraphicsMode Enumeration	37
2.1.17	HatchStyle Enumeration	38
2.1.18	ICMMode Enumeration	39
2.1.19	Illuminant Enumeration	39
2.1.20	Letterform Enumeration	40
2.1.21	MapMode Enumeration	41
2.1.22	MetafileVersion Enumeration	42
2.1.23	MidLine Enumeration	42
2.1.24	ModifyWorldTransformMode Enumeration	43
2.1.25	PenStyle Enumeration	43
2.1.26	Point Enumeration	45
2.1.27	PolygonFillMode Enumeration	45
2.1.28	Proportion Enumeration	45
2.1.29	RegionMode Enumeration	46
2.1.30	SerifType Enumeration	47
2.1.31	StockObject Enumeration	48
2.1.32	StretchMode Enumeration	50
2.1.33	StrokeVariation Enumeration	51
2.1.34	Weight Enumeration	51

2.1.35	XHeight Enumeration	52
2.2	EMF Objects	53
2.2.1	BitFIX28_4 Object	53
2.2.2	ColorAdjustment Object	53
2.2.3	DesignVector Object	55
2.2.4	EmrFormat Object	55
2.2.5	EmrText Object	56
2.2.6	EpsData Object	58
2.2.7	GradientRectangle Object	59
2.2.8	GradientTriangle Object	59
2.2.9	Header Object	60
2.2.10	HeaderExtension1 Object	62
2.2.11	HeaderExtension2 Object	62
2.2.12	LogBrushEx Object	63
2.2.13	LogFont Object	63
2.2.14	LogFontEx Object	66
2.2.15	LogFontExDv Object	68
2.2.16	LogFontPanose Object	69
2.2.17	LogPalette Object	71
2.2.18	LogPaletteEntry Object	71
2.2.19	LogPen Object	72
2.2.20	LogPenEx Object	72
2.2.21	Panose Object	74
2.2.22	PixelFormatDescriptor Object	75
2.2.23	Point28_4 Object	78
2.2.24	RegionData Object	79
2.2.25	RegionDataHeader Object	79
2.2.26	TriVertex Object	80
2.2.27	UniversalFontId Object	81
2.2.28	XForm Object	82
2.3	EMF Records	83
2.3.1	Bitmap Record Types	83
2.3.1.1	EMR_ALPHABLEND Record	86
2.3.1.2	EMR_BITBLT Record	90
2.3.1.3	EMR_MASKBLT Record	93
2.3.1.4	EMR_PLGBLT Record	98
2.3.1.5	EMR_SETDIBITSTODEVICE Record	102
2.3.1.6	EMR_STRETCHBLT Record	104
2.3.1.7	EMR_STRETCHDIBITS Record	108
2.3.1.8	EMR_TRANSPARENTBLT Record	111
2.3.2	Clipping Record Types	114
2.3.2.1	EMR_EXCLUDECLIPRECT Record	115
2.3.2.2	EMR_EXTSELECTCLIPRGN Record	116
2.3.2.3	EMR_INTERSECTCLIPRECT Record	117
2.3.2.4	EMR_OFFSETCLIPRGN Record	117
2.3.2.5	EMR_SELECTCLIPPATH Record	118
2.3.3	Comment Record Types	118
2.3.3.1	EMR_COMMENT Record	120
2.3.3.2	EMR_COMMENT_EMFPLUS Record	120
2.3.3.3	EMR_COMMENT_EMFSPOOL Record	121
2.3.3.4	EMR_COMMENT_PUBLIC Record Types	122
2.3.3.4.1	EMR_COMMENT_BEGINGROUP Record	123
2.3.3.4.2	EMR_COMMENT_ENDGROUP Record	124

2.3.3.4.3	EMR_COMMENT_MULTIFORMATS Record	125
2.3.3.4.4	EMR_COMMENT_WINDOWS_METAFILE Record	126
2.3.4	Control Record Types	127
2.3.4.1	EMR_EOF Record	128
2.3.4.2	EMR_HEADER Record Types	129
2.3.4.2.1	EmfMetafileHeader Record	133
2.3.4.2.2	EmfMetafileHeaderExtension1 Record	134
2.3.4.2.3	EmfMetafileHeaderExtension2 Record	136
2.3.5	Drawing Record Types	139
2.3.5.1	EMR_ANGLEARC Record	143
2.3.5.2	EMR_ARC Record	144
2.3.5.3	EMR_ARCTO Record	145
2.3.5.4	EMR_CHORD Record	146
2.3.5.5	EMR_ELLIPSE Record	147
2.3.5.6	EMR_EXTFLOODFILL Record	147
2.3.5.7	EMR_EXTTEXTOUTA Record	148
2.3.5.8	EMR_EXTTEXTOUTW Record	149
2.3.5.9	EMR_FILLPATH Record	150
2.3.5.10	EMR_FILLRGN Record	151
2.3.5.11	EMR_FRAMERGN Record	152
2.3.5.12	EMR_GRADIENTFILL Record	153
2.3.5.13	EMR_LINETO Record	155
2.3.5.14	EMR_PAINTRGN Record	155
2.3.5.15	EMR_PIE Record	156
2.3.5.16	EMR_POLYBEZIER Record	157
2.3.5.17	EMR_POLYBEZIER16 Record	158
2.3.5.18	EMR_POLYBEZIERTO Record	159
2.3.5.19	EMR_POLYBEZIERTO16 Record	160
2.3.5.20	EMR_POLYDRAW Record	161
2.3.5.21	EMR_POLYDRAW16 Record	162
2.3.5.22	EMR_POLYGON Record	163
2.3.5.23	EMR_POLYGON16 Record	164
2.3.5.24	EMR_POLYLINE Record	165
2.3.5.25	EMR_POLYLINE16 Record	166
2.3.5.26	EMR_POLYLINETO Record	167
2.3.5.27	EMR_POLYLINETO16 Record	168
2.3.5.28	EMR_POLYPOLYGON Record	169
2.3.5.29	EMR_POLYPOLYGON16 Record	170
2.3.5.30	EMR_POLYPOLYLINE Record	171
2.3.5.31	EMR_POLYPOLYLINE16 Record	172
2.3.5.32	EMR_POLYTEXTOUTA Record	174
2.3.5.33	EMR_POLYTEXTOUTW Record	175
2.3.5.34	EMR_RECTANGLE Record	176
2.3.5.35	EMR_ROUNDRECT Record	177
2.3.5.36	EMR_SETPIXELV Record	177
2.3.5.37	EMR_SMALLTEXTOUT Record	178
2.3.5.38	EMR_STROKEANDFILLPATH Record	180
2.3.5.39	EMR_STROKEPATH Record	180
2.3.6	Escape Record Types	181
2.3.6.1	EMR_DRAWESCAPE Record	182
2.3.6.2	EMR_EXTESCAPE Record	183
2.3.6.3	EMR_NAMEDESCAPE Record	183
2.3.7	Object Creation Record Types	184

2.3.7.1	EMR_CREATEBRUSHINDIRECT Record	186
2.3.7.2	EMR_CREATECOLORSPACE Record	187
2.3.7.3	EMR_CREATECOLORSPACEW Record	188
2.3.7.4	EMR_CREATEDIBPATTERNBRUSHPT Record	189
2.3.7.5	EMR_CREATEMONOBRUSH Record	190
2.3.7.6	EMR_CREATEPALETTE Record	192
2.3.7.7	EMR_CREATEPEN Record	193
2.3.7.8	EMR_EXTCREATEFONTINDIRECTW Record	193
2.3.7.9	EMR_EXTCREATEPEN Record	195
2.3.8	Object Manipulation Record Types	197
2.3.8.1	EMR_COLORCORRECTPALETTE Record	198
2.3.8.2	EMR_DELETECOLORSPACE Record	199
2.3.8.3	EMR_DELETEOBJECT Record	200
2.3.8.4	EMR_RESIZEPALETTE Record	200
2.3.8.5	EMR_SELECTOBJECT Record	201
2.3.8.6	EMR_SELECTPALETTE Record	201
2.3.8.7	EMR_SETCOLORSPACE Record	202
2.3.8.8	EMR_SETPALETTEENTRIES Record	203
2.3.9	OpenGL Record Types	203
2.3.9.1	EMR_GLSBOUNDEDRECORD Record	205
2.3.9.2	EMR_GLSRECORD Record	205
2.3.10	Path Bracket Record Types	206
2.3.11	State Record Types	207
2.3.11.1	EMR_COLORMATCHTOTARGETW Record	211
2.3.11.2	EMR_FORCEUFIMAPPING Record	212
2.3.11.3	EMR_INVERTRGN Record	213
2.3.11.4	EMR_MOVETOEX Record	214
2.3.11.5	EMR_PIXELFORMAT Record	214
2.3.11.6	EMR_RESTOREDC Record	215
2.3.11.7	EMR_SCALEVIEWPORTEXTEX Record	216
2.3.11.8	EMR_SCALEWINDOWEXTEX Record	216
2.3.11.9	EMR_SETARCDIRECTION Record	217
2.3.11.10	EMR_SETBKCOLOR Record	218
2.3.11.11	EMR_SETBKMODE Record	219
2.3.11.12	EMR_SETBRUSHORGEX Record	219
2.3.11.13	EMR_SETCOLORADJUSTMENT Record	220
2.3.11.14	EMR_SETICMMODE Record	220
2.3.11.15	EMR_SETICMPROFILEA Record	221
2.3.11.16	EMR_SETICMPROFILEW Record	222
2.3.11.17	EMR_SETLAYOUT Record	223
2.3.11.18	EMR_SETLINKEDUFIS Record	223
2.3.11.19	EMR_SETMAPMODE Record	224
2.3.11.20	EMR_SETMAPPERFLAGS Record	225
2.3.11.21	EMR_SETMITERLIMIT Record	225
2.3.11.22	EMR_SETPOLYFILLMODE Record	226
2.3.11.23	EMR_SETROP2 Record	226
2.3.11.24	EMR_SETSTRETCHBLTMODE Record	227
2.3.11.25	EMR_SETTEXTALIGN Record	228
2.3.11.26	EMR_SETTEXTCOLOR Record	228
2.3.11.27	EMR_SETTEXTJUSTIFICATION Record	229
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
3 Structure Examples	235
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
3.2 EMF Metafile Example.....	237
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	256
3.2.5 EMR_SELECTOBJECT Example.....	258
3.2.6 EMR_BITBLT Example	259
3.2.7 EMR_SETBKMODE Example	274
3.2.8 EMR_EXTCREATEFONTINDIRECTW Example	274
3.2.9 EMR_SELECTOBJECT Example.....	277
3.2.10 EMR_EXTEXTOUTW Example	278
3.2.11 EMR_EXTCREATEFONTINDIRECTW Example	279
3.2.12 EMR_SELECTOBJECT Example.....	282
3.2.13 EMR_EXTCREATEFONTINDIRECTW Example	283
3.2.14 EMR_SELECTOBJECT Example	286
3.2.15 EMR_DELETEOBJECT Example.....	286
3.2.16 EMR_EXTCREATEFONTINDIRECTW Example	287
3.2.17 EMR_SELECTOBJECT Example.....	289
3.2.18 EMR_SELECTOBJECT Example.....	290
3.2.19 EMR_DELETEOBJECT Example.....	290
3.2.20 EMR_DELETEOBJECT Example.....	291
3.2.21 EMR_SELECTOBJECT Example	291
3.2.22 EMR_EOF Example.....	291
4 Security Considerations.....	293
5 Appendix A: Product Behavior.....	294
6 Change Tracking.....	302
7 Index	304

1 Introduction

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

An EMF metafile is a series of variable-length records, called EMF records, which contain graphics drawing commands, object definitions, and properties. The metafile begins with a header record, which includes the metafile version, its size, the resolution of the device on which the picture was created, and the dimensions of the picture. An EMF 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, and desktops, or in the client areas of applications.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

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

American National Standards Institute (ANSI) character set

ASCII

big-endian

color profile

enhanced metafile format (EMF)

enhanced metafile format plus extensions (EMF+)

enhanced metafile spool format (EMFSPOOL)

Graphics Device Interface (GDI)

Graphics Device Interface, Extended (GDI+)

Image Color Management (ICM)

little-endian

original equipment manufacturer (OEM) character set

PostScript

print job

print server

printer driver

spool file

Unicode

UTF-16LE (Unicode Transformation Format, 16 bits, little-endian)

Windows metafile format (WMF)

The following terms are specific to this document:

28.4 bit FIX notation: A notation for representing the location of a point on a device surface to within one-sixteenth of a pixel. Each point coordinate is a 32-bit value, of which the 28 higher-order bits are the signed integral part and the 4 lower-order bits are the unsigned fractional part, in one-sixteenth units of distance.

For example, the number 0x0000003C is a coordinate value of 3.75, because the fractional part is 12 sixteenths, or 0.75.

additive color model: A **color model** that 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.

anti-aliasing: The smoothing of the jagged appearance of font characters and lines, which is an artifact of the limited resolution on an output device. The pixels that surround the edges of the character glyph or line are changed to varying shades of color in order to blend the sharp edge into the background.

ascent: The distance that characters of a font **typeface** extend above the top of a lowercase "x".

aspect ratio: The ratio that is computed by dividing the width of a pixel on a given output device by its height.

baseline: The imaginary line to which the bottom of the lowercase "x" character in a font **typeface** is aligned.

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 contains a device-independent representation of a graphical image, a **logical palette**, dimensions, and other information.

brightness: The relative lightness or darkness of an image, or of a particular color in an image.

cell height: A vertical measure of font size, which is the sum of the font height and **internal leading**. It might not be the same as the distance between two lines of text.

color channel: A component color from which all colors in an image are rendered. In an **RGB color space**, there are color channels for red, green, and blue. In a grayscale **color space**, the color channels are black and white. Color channel values typically range from 0 to 255.

color correction: Altering the colors in an image in order to print or display it such that the colors correctly match reality.

colorfulness: A concept referring to the perceived **intensity** of a specific color, the difference between a color against gray.

color gamut: The entire range of colors that is 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 plane: In **bitmap** graphics, all pixel information for a single color. See **color channel**.

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 multidimensional coordinate system. The number of dimensions is generally two, three, or four. For example, grayscales, which are combinations of only black and white, can be mapped to a two-dimensional color space. 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. If transparency is considered one of the components of an **RGB** color, four dimensions are appropriate.

color table: An array of data that maps pixel values into a **color space**.

compositing: The process that takes place during image rendering, which combines color data from multiple graphics **region**.

contrast: The relative difference between lightness and darkness in an area of an image.

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 vector: A set of specific values for the properties 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 ([\[MS-WMF\]](#) section 2.2.2.9).

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.

diacritic: A small sign such as an accent mark that is added to a letter to alter pronunciation or to distinguish between similar usages.

dithering: A digital representation of continuous-tone graphics using **halftones**.

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

em size: A measure of font size, which is the **cell height** minus the **internal leading**. An "em" is a term that has been used historically as a unit of typeset size.

encapsulated PostScript (EPS): A file of **PostScript** raw data that describes the appearance of a single page. EPS data can describe text, graphics, and images; but the primary purpose of an EPS file is to be encapsulated within another **PostScript** page definition.

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 are the possible values for the property.

font hinting: The use of mathematical operations to manipulate the appearance of an **outline font** so that it lines up with a **rasterized** grid. At small resolutions, with or without **anti-aliasing**, hinting is critical for producing clear, legible text for human readers.

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 affects midrange tones.

gamma correction: An adjustment to the light **intensity** (brightness) of a graphics device in order to match the output more closely to the original image.

halftone: A color representation consisting of a discrete gray or tone level.

inclusive-inclusive: When referring to the bounds of a rectangle that consists of two coordinates—one coordinate for one corner and the other coordinate for the opposite corner—the coordinates are considered 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.

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

internal leading: The amount of space inside a character cell, within the bounds set by the font **ascent**. Accent marks and other **diacritics** can occur in this area.

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) [\[JFIF\]](#), and their file extensions are .JPG or .JFF.

line cap: The shape to use at the end of a line drawn by a graphics pen.

line join: The shape to use at the intersection of two lines drawn by a graphics pen.

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-axis and y-axis 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. 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.

miter length: At the intersection of two lines, the distance from the intersection of the line walls on the inside of the **line join** to the intersection of the line walls on the outside of the **line join**. The miter length can be large when the angle between two lines is small. If the miter length of the join of an intersection exceeds a specified limit, the join can be beveled to keep it within the limit of the join of the intersection.

monoscopic: The property of an image that conveys a lack of the illusion of depth, as if the image were two-dimensional.

multiple master: A font technology that is a variation of the **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 that supports the rendering of multidimensional graphical objects. The Windows implementation of OpenGL ([OPENGL](#)) is industry-standard graphics software with which implementers can create high-quality still and animated three-dimensional color images.

OpenType: A **Unicode**-based font technology that is an extension to **TrueType** and **Type 1 font** technologies. OpenType allows **PostScript** and **TrueType** glyph definitions to reside in a common container format.

outline font: A font that is defined with mathematical equations, which makes it possible for a printer or other output device to generate the characters at any size. Besides being arbitrarily scalable, the appearance of an outline font improves in proportion to the resolution of an output device. **TrueType** and **PostScript** are examples of outline font technology.

packed DIB: A **device-independent bitmap (DIB)** in which the bit array immediately follows the header ([MS-WMF](#) section 2.2.2.9).

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.

path: A graphics object that is a container for a series of line and curve segments and **regions** in an image.

path bracket: A series of **paths** that composes a larger figure. A path bracket specifies the current **path** that is defined in the **playback device context**.

pitch: A property of a font that describes the horizontal density of characters in a font; that is, the number of characters that can fit in a given unit of space. When all the characters in a font have the same width, the font is called "fixed-pitch"; if characters can have various widths, the font is "variable-pitch".

"Times New Roman" is a variable-pitch font; it is easy to see that the characters in the font may have different widths. For example, the width of a lowercase "i" is visibly less than the width of an uppercase "W".

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.

PNG: Portable Network Graphics (PNG): A bitmapped graphics file format [\[RFC2083\]](#) 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.

rasterized font: A font produced with matrixes of discrete pixel settings. Such fonts are not scalable, but must define glyph **bitmaps** at specific sizes. Because of this, the appearance of rasterized fonts does not improve in proportion to the resolution of an output device and, when magnified, appear significantly worse than **vector fonts**.

rasterizer: A program that converts geometric shapes into matrixes of discrete pixel settings on a graphics object such as a font.

reflection transform: A **transform** that is used to create a mirror image of an object with respect to either the horizontal or vertical axis.

region: A graphics object that is nonrectilinear in shape and is defined by an array of scanlines.

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

RGBA: A **color space** in which each pixel is a group of four components in the following order: red, green, blue, **alpha**. The red, green, and blue values specify the **intensity** of each color; the values range from 0.0 (least intense) to 1.0 (most intense). The number of bits for each component varies depends on the pixel format. The color displayed is a result of the sum of the three color values. If all three values are 0.0, the result is black. If all three values are 1.0, the result is white.

rotation transform: A **transform** that is used to rotate an object. When rotation occurs, the points that make up an object are rotated with respect to the **coordinate space** origin.

scaling transform: A **transform** that is used to stretch or compress an object horizontally or vertically.

shear transform: A **transform** that is used to shear or cut an object. There are two components of a shear **transform**: The first alters the vertical lines in an object, and the second alters the horizontal lines.

stereoscopic: The property of an image that gives the illusion of depth, as if the image were three-dimensional. The pixels that compose such an image can include a **color plane** that is designed to add that illusion.

stock object: A predefined graphics object. Stock objects are standard, commonly-used objects, such as a black brush and pen. The set of predefined stock objects is specified in the [StockObject enumeration \(section 2.1.31\)](#). Stock objects are neither created nor deleted.

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**.

tint: The amount of a neutral color, such as black or white, which is mixed with another color. Changing the tint increases or decreases the **intensity** and saturation, and leaves the coordinates of the color in the **color space** unchanged.

transform: 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.

transformation: See **transform**.

translation transform: A **transform** that is used to shift each point in an object vertically, horizontally, or both, by a specified amount.

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 its design. Both TrueType and **Type 1 font** technologies are part of the **OpenType** format.

Type 1 font: A public, standard type format originally developed 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 primary design of a set of printed characters such as **Courier**, **Helvetica**, and **Times Roman**.

The terms typeface and font are sometimes used interchangeably. 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 font: A font that is defined with geometrical primitives such as points, lines, curves, and polygons, which are all based on mathematical equations instead of collections of discrete pixel settings. Vector fonts can be rendered in high quality at arbitrary sizes. **Outline fonts** are vector fonts. Contrast with **rasterized fonts**.

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

Windows Color System (WCS): A superset of **ICM** APIs and functionality.[<1>](#)

world space: The most abstract logical **coordinate space** for graphics **transform**.

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=28245

Note There is a charge to download the specification.

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

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

[RFC2083] Boutell, T., "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.rfc-editor.org/rfc/rfc2119.txt>

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

1.2.2 Informative References

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

[MS-EMFSPOOL] Microsoft Corporation, "[Enhanced Metafile Spool Format](#)".

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

[MSDN-GDI+] Microsoft Corporation, "GDI+", <http://msdn.microsoft.com/en-us/library/ms533798.aspx>

[MSDN-WRLDPGSPC] Microsoft Corporation, "World-Space to Page-Space Transformations", <http://msdn.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>

1.3 Overview

1.3.1 Metafile Structure

The 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 metafiles are:

- **Original:** This is the first version of EMF metafile, 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.

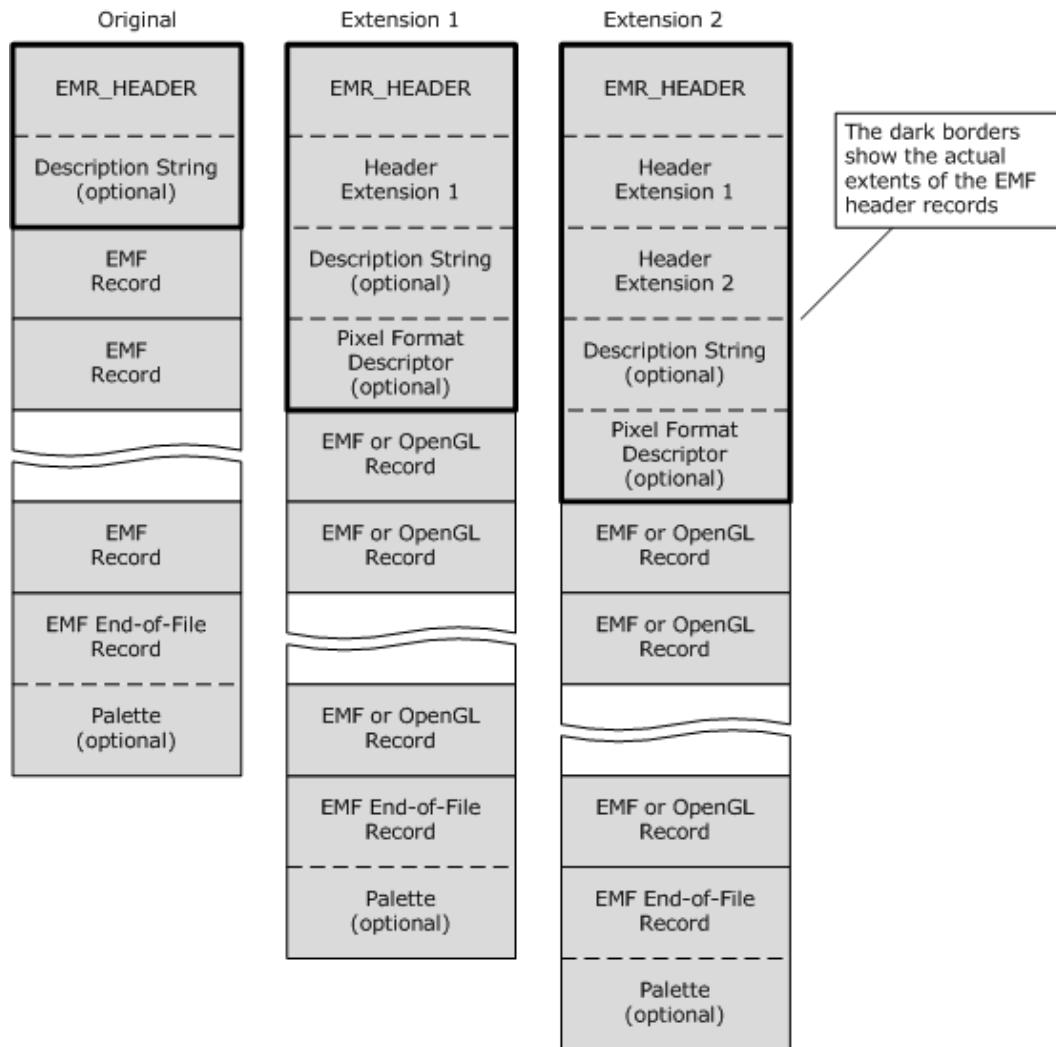


Figure 1: EMF metafile structures

Thus, all EMF metafiles 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 contains drawing orders, graphics state information, and graphics object definitions. At least one record has to 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 has to 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 \(section 2.2.18\)](#) 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 are part of the EMF header record, and cannot be located at some arbitrary location in the EMF metafile, such as between EMF records.

EMF records are 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 is used.

1.3.2 Graphics Objects

Graphics objects, which are used in the drawing and painting operations specified in the records of an EMF 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 be 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 [2.2.12](#)
- Color spaces, specified in [\[MS-WMF\]](#) sections [2.2.2.11](#) and [2.2.2.12](#)
- Fonts, specified in sections [2.2.13](#), [2.2.14](#), [2.2.15](#), and [2.2.16](#)
- Palettes, specified in section [2.2.17](#)
- Pens, specified in sections [2.2.19](#) and [2.2.20](#)

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 [2.1.31](#).

1.3.3 Byte Ordering

Data in metafile records is 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 byte numbering used for bitfields in this specification is big-endian. Other architectures number the bytes in a binary word from right to left, which is referred to as little-endian. The byte numbering used for enumerations, objects, and records in this specification is little-endian.

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

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

EMF is related to the following file formats:

- **Windows metafile format (WMF)** [\[MS-WMF\]](#) uses similar graphics, commands, and objects, but WMF files are not device-independent.[<5>](#)
- **Enhanced metafile spool format (EMFSPOOL)** [\[MS-EMFSPOOL\]](#) is an application of EMF for **print job** spooling.
- **Enhanced metafile format plus extensions (EMF+)** [\[MS-EMFPLUS\]](#) specifies object-oriented structures that can be embedded in EMF metafiles.[<6>](#)

1.5 Applicability Statement

Files that adhere to the 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 specification covers versioning issues in the following areas:

Versioning: The EMF structure has been revised twice. The different versions are:

- Original: The first version of the EMF structure, supporting records that define drawing commands and graphics objects.

- 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

EMF metafiles define a mechanism for the encapsulation of arbitrary vendor-defined data. The record type [EMR_COMMENT](#) can contain arbitrary private data that is unknown to EMF. This data is meaningful only to applications that can decode the format of the data.

2 Structures

The following sections specify various types of EMF 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 **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_SETWINDOWEXTEX = 0x00000009,
    EMR_SETWINDOWORGEX = 0x0000000A,
    EMR_SETVIEWPORTEXTEX = 0x0000000B,
    EMR_SETVIEWPORTORGEX = 0x0000000C,
    EMR_SETBRUSHORGEX = 0x0000000D,
    EMR_EOF = 0x0000000E,
    EMR_SETPIXELV = 0x0000000F,
    EMR_SETMAPPERFLAGS = 0x00000010,
    EMR_SETMAPMODE = 0x00000011,
    EMR_SETBKMODE = 0x00000012,
    EMR_SETPOLYFILLCODE = 0x00000013,
    EMR_SETROP2 = 0x00000014,
    EMR_SETSTRETCHBLT = 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_SCALEVIEWPORTEXTEX = 0x0000001F,
    EMR_SCALEWINDOWEXTEX = 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_ROUNIRECT = 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_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_EXTEXTOUTA = 0x00000053,
EMR_EXTEXTOUTW = 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_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_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_SETWINDOWEXTEX: This record defines the window extent.

EMR_SETWINDOWORGEX: This record defines the window origin.

EMR_SETVIEWPORTEXTEX: This record defines the viewport extent.

EMR_SETVIEWPORTORIGIN: This record defines the viewport origin.

EMR_SETBRUSHORIGIN: This record defines the origin of the current brush.

EMR_EOF: This record indicates the end of the metafile.

EMR_SETPIXEL: This record defines the color of the pixel at the specified logical coordinates.

EMR_SETMAPPERFLAGS: This record specifies parameters of the process of matching logical fonts to physical fonts, which is performed by the **font mapper**.[<9>](#)

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-axis and y-axis.

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_SCALEVIEWPORTEXTEX: This record redefines the viewport for the playback device context using the ratios formed by the specified multiplicands and divisors.

EMR_SCALEWINDOWEXTEX: 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 \(section 2.3.11\)](#) 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 \(section 3.1.1.1\)](#).

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 for figure filling 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 \(section 2.2.17\)](#) 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_COMMENT: This record specifies arbitrary private data.

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 specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation.

EMR_STRETCHBLT: This record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

EMR_MASKBLT: This record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern and with the application of a color mask bitmap, according to specified foreground and background raster operations.

EMR_PLGBLT: This record specifies a block transfer of pixels from a source bitmap to a destination parallelogram, with the application of a color mask bitmap.

EMR_SETDIBITSTODEVICE: This record specifies a block transfer of pixels from specified scanlines of a source bitmap to a destination rectangle.

EMR_STRETCHDIBITS: This record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, 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_EXTEXTOUTA: This record draws an **ASCII** text string using the current font and text colors.

Note [EMR_EXTEXTOUTA](#) SHOULD be emulated with an [EMR_EXTEXTOUTW](#) record (section 2.3.5.8). [<10>](#) This requires the ASCII text string in the [EmrText](#) object to be converted to Unicode UTF16-LE encoding.

EMR_EXTEXTOUTW: This record draws a Unicode text string using the current font and text colors.

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_POLYBEZIER16: 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 **device-independent bitmap (DIB)** section bitmap or it can be a device-dependent bitmap.

EMR_CREATEDIBPATTERNBRUSHPT: This record defines a logical brush that has the pattern specified by the 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 draws one or more ASCII text strings using the current font and text colors.

Note [EMR_POLYTEXTOUTA](#) SHOULD be emulated with a series of EMR_EXTEXTOUTW records, one per string.[<11>](#)

EMR_POLYTEXTOUTW: This record draws one or more Unicode text strings using the current font and text colors.

Note [EMR_POLYTEXTOUTW](#) SHOULD be emulated with a series of EMR_EXTEXTOUTW records, one per string.[<12>](#)

EMR_SETICMMODE: This record specifies the mode of **Image Color Management (ICM)** for graphics operations.[<13>](#)

EMR_CREATECOLORSPACE: This record creates a logical **color space** object from a **color profile** with a name consisting of ASCII characters.[<14>](#)

EMR_SETCOLORSPACE: This record defines the current logical color space object for graphics operations.[<15>](#)

EMR_DELETECOLORSPACE: This record deletes a logical color space object.[<16>](#)

Note An [EMR_DELETEOBJECT](#) record SHOULD be used instead of [EMR_DELETECOLORSPACE](#) to delete a logical color space object.[<17>](#)

EMR_GLSRECORD: This record specifies an OpenGL function.[<18>](#)

EMR_GLSBOUNDEDRECORD: This record specifies an OpenGL function with a bounding rectangle for output.[<19>](#)

EMR_PIXELFORMAT: This record specifies the pixel format to use for graphics operations.[<20>](#)

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_SMALLTEXTOUT: This record outputs a string.

EMR_FORCEUFIIMAPPING: 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 specifies how to correct the entries of a logical palette object using **Windows Color System (WCS)** 1.0 values.[<21>](#)

EMR_SETICMPROFILEA: This record specifies a color profile in a file with a name consisting of ASCII characters, for graphics output.[<22>](#)

EMR_SETICMPROFILEW: This record specifies a color profile in a file with a name consisting of Unicode characters, for graphics output.[<23>](#)

EMR_ALPHABLEND: This record specifies a block transfer of pixels from a source bitmap to a destination rectangle, including **alpha transparency** data, according to a specified blending operation.[<24>](#)

EMR_SETLAYOUT: This record specifies the order in which text and graphics are drawn.[<25>](#)

EMR_TRANSPARENTBLT: This record specifies a block transfer of pixels from a source bitmap to a destination rectangle, treating a specified color as transparent, stretching or compressing the output to fit the dimensions of the destination, if necessary.[<26>](#)

EMR_GRADIENTFILL: This record specifies filling rectangles or triangles with gradients of color.[<27>](#)

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

EMR_SETTEXTJUSTIFICATION: This record specifies the amount of extra space to add to break characters for justification purposes.[<28>](#)

EMR_COLORMATCHTOTARGETW: This record specifies whether to perform **color matching** with a color profile that is specified in a file with a name consisting of Unicode characters.[<29>](#)

EMR_CREATECOLORSPACEW: This record creates a logical color space object from a color profile with a name consisting of Unicode characters.[<30>](#)

2.1.2 ArcDirection Enumeration

The **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 **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: Nonstraight arms/horizontal.

PAN_BENT_ARMS_WEDGE: Nonstraight arms/wedge.

PAN_BENT_ARMS_VERT: Nonstraight arms/vertical.

PAN_BENT_ARMS_SINGLE_SERIF: Nonstraight arms/single-serif.

PAN_BENT_ARMS_DOUBLE_SERIF: Nonstraight arms/double-serif.

2.1.4 BackgroundMode Enumeration

The **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 **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 **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 **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 **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 **DIBColors** enumeration defines how to interpret the values in the color table of a DIB.

```
typedef enum
{
    DIB_RGB_COLORS = 0x00,
    DIB_PAL_COLORS = 0x01,
    DIB_PAL_INDICES = 0x02
} 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 [LogPalette](#) object (section [2.2.17](#)) that is currently defined in the playback device context.

DIB_PAL_INDICES: No color table exists. The pixels in the DIB are indices into the current logical palette in the playback device context.

DIBs are specified by [DeviceIndependentBitmap](#) objects ([\[MS-WMF\]](#) section 2.2.2.9).

2.1.10 EmrComment Enumeration

The **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 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 an 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 **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. [<31>](#)

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. [<32>](#)

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. [<33>](#)

ETO_NUMERICSLATIN: This bit indicates that to display numbers, European digits SHOULD be used. [<34>](#)

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. [<35>](#)

ETO_PDY: This bit indicates that both horizontal and vertical character displacement values SHOULD be provided.[<36>](#)

ETO_REVERSE_INDEX_MAP: This bit is reserved and SHOULD NOT be used.[<37>](#)

2.1.12 FamilyType Enumeration

The **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 **FloodFill** enumeration defines values that specify how to determine the area for a flood fill operation.

```
typedef enum
{
    FLOODFILLBORDER = 0x00000000,
    FLOODFILLSURFACE = 0x00000001
} FloodFill;
```

FLOODFILLBORDER: The fill area is bounded by a specific color.

FLOODFILLSURFACE: The fill area is defined by a specific color. 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 **FormatSignature** enumeration defines values that are used to identify the format of embedded data in EMF records.

```
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 EMF record data.

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

This signature is used in an [EMR_HEADER record \(section 2.3.4.2\)](#) to identify the EMF metafile, and it is used in an [EmrFormat object \(section 2.2.4\)](#) to specify embedded EMF record data in an [EMR_COMMENT_MULTIFORMATS record \(section 2.3.3.4.3\)](#).

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 **encapsulated PostScript (EPS)** format data.

This signature is used in an EmrFormat object to specify embedded PostScript data in an [EpsData object \(section 2.2.6\)](#) in an EMR_COMMENT_MULTIFORMATS record.

2.1.15 GradientFill Enumeration

The **GradientFill** enumeration defines the modes for gradient fill operations.

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

GRADIENT_FILL_RECT_H: A mode in which color interpolation is performed along a gradient from the left to the right edges of a rectangle.

GRADIENT_FILL_RECT_V: A mode in which color interpolation is performed along a gradient from the top to the bottom edges of a rectangle.

GRADIENT_FILL_TRIANGLE: A mode in which color interpolation is performed between vertexes of a triangle.

2.1.16 GraphicsMode Enumeration

The **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 are rotated about the x-axis 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, which might require a rotation along the x-axis 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^{<38>} 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^{<39>} 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 **HatchStyle** enumeration is an extension to the WMF **HatchStyle** enumeration ([\[MS-WMF\] section 2.1.1.12](#)).

```
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 **ICMMode** enumeration defines values that specify when to turn on and off ICM.

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

ICM_OFF: Turns off Image Color Management (ICM) in the playback device context. Turns on old-style color correction of halftones.[<40>](#)

ICM_ON: Turns on ICM in the playback device context. Turns off old-style color correction of halftones.[<41>](#)

ICM_QUERY: Queries the current state of color management in the playback device context.[<42>](#)

ICM_DONE_OUTSIDEDC: Turns off ICM in the playback device context, and turns off old-style color correction of halftones.[<43>](#)

2.1.19 Illuminant Enumeration

The **Illuminant** enumeration defines values that specify the illuminant value of an image, which determines 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 **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 **MapMode** enumeration is used to define the unit of measure for transforming page space units into device space units and for defining 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_SETWINDOWEXTEX](#) and [EMR_SETVIEWPORTEXTEX](#) records SHOULD be used to specify the units and the orientation of the axes.

Adjustments MUST be made as necessary to ensure that 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_SETWINDOWTEX and EMR_SETVIEWPORTEXTEX records SHOULD be used to specify the units, orientation, and scaling.

2.1.22 MetafileVersion Enumeration

The **MetafileVersion** enumeration defines the interoperability version for EMF metafile.

```
typedef enum
{
    META_FORMAT_ENHANCED = 0x00010000
} MetafileVersion;
```

META_FORMAT_ENHANCED: Specifies EMF metafile interoperability.

2.1.23 MidLine Enumeration

The **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 **ModifyWorldTransformMode** enumeration defines modes for using specified transform data to modify the world-space to page-space transform that is currently defined in the playback device context.

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

MWT_IDENTITY: Reset the current transform using the identity matrix. In this mode, the specified transform data is ignored.

MWT_LEFTMULTIPLY: Multiply the current transform. In this mode, the specified transform data is the left multiplicand, and the transform that is currently defined in the playback device context is the right multiplicand.

MWT_RIGHTMULTIPLY: Multiply the current transform. In this mode, the specified transform data is the right multiplicand, and the transform that is currently defined in the playback device context is the left multiplicand.

MWT_SET: Perform the function of an [EMR_SETWORLDTRANSFORM](#) record (section [2.3.12.2](#)).

A transform is specified in the form of an [XForm](#) object (section [2.2.28](#)). The modes specified by this enumeration apply to [EMR_MODIFYWORLDTRANSFORM](#) records (section [2.3.12.1](#)).

For more information concerning transforms and **coordinate spaces**, see [\[MSDN-WRLDPGSPC\]](#).

2.1.25 PenStyle Enumeration

The **PenStyle** enumeration defines the attributes of pens that can be used in graphics operations. A pen style is a combination of pen type, line style, **line cap**, and **line join**.

```
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: A pen type that specifies a line with a width of one logical unit and a style that is a solid color.

PS_ENDCAP_ROUND: A line cap that specifies round ends.

PS_JOIN_ROUND: A line join that specifies round joins.

PS_SOLID: A line style that is a solid color.

PS_DASH: A line style that is dashed.

PS_DOT: A line style that is dotted.

PS_DASHDOT: A line style that consists of alternating dashes and dots.

PS_DASHDOTDOT: A line style that consists of dashes and double dots.

PS_NULL: A line style that is invisible.

PS_INSIDEFRAME: A line style that is a solid color. When this style is specified in a 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: A line style that is defined by a styling array, which specifies the lengths of dashes and gaps in the line.

PS_ALTERNATE: A line style in which every other pixel is set. This style is applicable only to a pen type of **PS_COSMETIC**.

PS_ENDCAP_SQUARE: A line cap that specifies square ends.

PS_ENDCAP_FLAT: A line cap that specifies flat ends.

PS_JOIN_BEVEL: A line join that specifies beveled joins.

PS_JOIN_MITER: A line join that specifies mitered joins when the lengths of the joins are within the current **miter length** limit that is set in the playback device context. If the lengths of the joins exceed the miter limit, beveled joins are specified.

The miter length limit is a metafile state property that is set by the [EMR_SETMITERLIMIT](#) record.

PS_GEOEMTRIC: A pen type that specifies a line with a width that is measured in logical units and a style that can contain any of the attributes of a brush.

2.1.26 Point Enumeration

The **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 **PolygonFillMode** enumeration defines values that specify how to calculate the region of a polygon that 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 **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 **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 **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 **StockObject** enumeration specifies the indexes of predefined logical graphics objects that can be used in graphics operations.

The specific structures of stock objects are implementation-dependent; however, the properties of stock objects SHOULD be equivalent to the properties of explicitly created objects of the same type. These properties are specified where possible for the stock objects defined in this enumeration.

```
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: A white, solid-color brush that is equivalent to a logical brush ([LogBrushEx object](#), section 2.2.12) with the following properties:

- **BrushStyle:** **BS_SOLID** (WMF [BrushStyle enumeration](#), [\[MS-WMF\]](#) section 2.1.1.4)
- **Color:** 0x00FFFFFF (WMF [ColorRef object](#), [\[MS-WMF\]](#) section 2.2.2.8)

LTGRAY_BRUSH: A light gray, solid-color brush that is equivalent to a logical brush with the following properties:

- **BrushStyle:** **BS_SOLID**
- **Color:** 0x00C0C0C0

GRAY_BRUSH: A gray, solid-color brush that is equivalent to a logical brush with the following properties:

- **BrushStyle:** **BS_SOLID**

- **Color:** 0x00808080

DKGRAY_BRUSH: A dark gray, solid color brush that is equivalent to a logical brush with the following properties:

- **BrushStyle:** **BS_SOLID**
- **Color:** 0x00404040

BLACK_BRUSH: A black, solid color brush that is equivalent to a logical brush with the following properties:

- **BrushStyle:** **BS_SOLID**
- **Color:** 0x00000000

NULL_BRUSH: A null brush that is equivalent to a logical brush with the following properties: [<44>](#)

- **BrushStyle:** **BS_NULL**

WHITE_PEN: A white, solid-color pen that is equivalent to a logical pen ([LogPen object](#), section [2.2.19](#)) with the following properties:

- **PenStyle:** **PS_COSMETIC + PS_SOLID** ([PenStyle enumeration](#), section [2.1.25](#))
- **ColorRef:** 0x00FFFFFF (WMF ColorRef object).

BLACK_PEN: A black, solid-color pen that is equivalent to a logical pen with the following properties:

- **PenStyle:** **PS_COSMETIC + PS_SOLID**
- **ColorRef:** 0x00000000

NULL_PEN: A null pen that is equivalent to a logical pen with the following properties:

- **PenStyle:** **PS_NULL**

OEM_FIXED_FONT: A fixed-width, **OEM character set** font that is equivalent to a logical font ([LogFont object](#), section [2.2.13](#)) with the following properties:

- **Charset:** **OEM_CHARSET** (WMF [CharacterSet enumeration](#), [\[MS-WMF\]](#) section 2.1.1.5)
- **PitchAndFamily:** **FF_DONTCARE** (WMF [FamilyFont enumeration](#), [\[MS-WMF\]](#) section 2.1.1.8) + **FIXED_PITCH** (WMF [PitchFont enumeration](#), [\[MS-WMF\]](#) section 2.1.1.24)

ANSI_FIXED_FONT: A fixed-width font that is equivalent to a logical font with the following properties: [<45>](#)

- **Charset:** **ANSI_CHARSET**
- **PitchAndFamily:** **FF_DONTCARE + FIXED_PITCH**

ANSI_VAR_FONT: A variable-width font that is equivalent to a logical font with the following properties: [<46>](#)

- **Charset:** **ANSI_CHARSET**

- **PitchAndFamily: FF_DONTCARE + VARIABLE_PITCH**

SYSTEM_FONT: A font that is guaranteed to be available in the operating system. The actual font that is specified by this value is implementation-dependent.[<47>](#)

DEVICE_DEFAULT_FONT: The default font that is provided by the graphics device driver for the current output device. The actual font that is specified by this value is implementation-dependent.[<48>](#)

DEFAULT_PALETTE: The default palette that is defined for the current output device. The actual palette that is specified by this value is implementation-dependent.[<49>](#)

SYSTEM_FIXED_FONT: A fixed-width font that is guaranteed to be available in the operating system. The actual font that is specified by this value is implementation-dependent.

DEFAULT_GUI_FONT: The default font that is used for user interface objects such as menus and dialog boxes. The actual font that is specified by this value is implementation-dependent.[<50>](#)

DC_BRUSH: The solid-color brush that is currently selected in the playback device context.[<51>](#)

DC_PEN: The solid-color pen that is currently selected in the playback device context.[<52>](#)

During metafile processing, stock object indexes can be used by [object manipulation records](#) (section [2.3.8](#)) to select objects into the playback device context in the same way as indexes of graphics objects that are explicitly created by [object creation records](#) (section [2.3.7](#)). The index of a stock object can be distinguished from the index of an explicit object by the value of the most-significant bit. If that bit is set, the object is a stock object; if the bit is clear, the object was created by a previous metafile record.

2.1.32 StretchMode Enumeration

The **StretchMode** enumeration is used to specify how color data is added to or removed from bitmaps that are stretched or compressed.[<53>](#)

```
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_SETBRUSHORGEX](#) record. If it fails to do so, brush misalignment can occur.

2.1.33 StrokeVariation Enumeration

The **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 Weight Enumeration

The **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.

2.1.35 XHeight Enumeration

The **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 BitFIX28_4 Object

The BitFIX28_4 object defines a numeric value in **28.4 bit FIX notation**.

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
IntValue																												FracValue						

IntValue (28 bits): The signed, integral part of the number.

FracValue (4 bits): The unsigned fractional part of the number, in units of one-sixteenth.

The floating-point number represented by this object is computed as follows:

```
IntValue + (FracValue / 16)
```

2.2.2 ColorAdjustment Object

The ColorAdjustment object defines values for adjusting the colors in source bitmaps in bit-block transfers. [<54>](#)

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
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. This MUST be 0x0018.

Values (2 bytes): A 16-bit unsigned integer that specifies how to prepare the output image.

This field can be set to NULL or to any combination of values in the [ColorAdjustment](#) enumeration (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, from the [Illuminant](#) enumeration (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. This value SHOULD be in the range from 2,500 to 65,000. [<55>](#) 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. This value SHOULD be in the range from 2,500 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. This value SHOULD be in the range from 2,500 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. This value SHOULD be in the range from zero to 4,000.

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. This value SHOULD be in the range from 6,000 to 10,000.

Contrast (2 bytes): A 16-bit signed integer that specifies the amount of [contrast](#) to be applied to the source object. This value SHOULD be in the range from -100 to 100. A value of zero 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. This value SHOULD be in the range from -100 to 100. A value of zero 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. This value SHOULD be in the range from -100 to 100. A value of zero 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. This value SHOULD be in the range from -100 to 100. Positive numbers adjust towards red and negative numbers adjust towards green. A value of zero means tint adjustment MUST NOT be performed.

The ColorAdjustment object is used in bit-block transfers performed by [EMR_STRETCHBLT](#) and [EMR_STRETCHDIBITS](#) records when the [StretchMode](#) enumeration (section [2.1.32](#)) value is **STRETCH_HALFTONE**. The color adjustment values can apply a color filter or lighten or darken an image.

An [EMR_SETCOLORADJUSTMENT](#) record (section [2.3.11.13](#)) sets the current ColorAdjustment object in the playback device context. That ColorAdjustment object affects all subsequent [EMR_STRETCHBLT](#) and [EMR_STRETCHDIBITS](#) records until a different ColorAdjustment object is specified by another [EMR_SETCOLORADJUSTMENT](#) record, or until the object is removed by a [EMR_DELETEOBJECT](#) record.

2.2.3 DesignVector Object

The DesignVector (section 2.2.3) 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	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
Signature																																		
NumAxes																																		
Values (variable)																																		
...																																		

Signature (4 bytes): A 32-bit unsigned integer that MUST be set to the value 0x08007664.

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.

2.2.4 EmrFormat Object

The EmrFormat object contains information that identifies the format of image data in an [EMR COMMENT MULTIFORMATS record \(section 2.3.3.4.3\)](#).

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
Signature																																		
Version																																		
SizeData																																		
offData																																		

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

Version (4 bytes): A 32-bit unsigned integer that specifies the format version number. If the **Signature** field specifies encapsulated PostScript (EPS), this value MUST be 0x00000001; otherwise, this value MUST be ignored.

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 **identifier** field in an [EMR COMMENT PUBLIC record \(section 2.3.3.4\)](#). The offset MUST be 32-bit aligned.

2.2.5 EmrText Object

The EmrText object contains values for text output.

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
Reference																																		
...																																		
Chars																																		
offString																																		
Options																																		
Rectangle																																		
...																																		
...																																		
...																																		
offDx																																		
StringBuffer (variable)																																		
...																																		
DxBuffer (variable)																																		
...																																		

Reference (8 bytes): A WMF [PointL](#) object ([\[MS-WMF\]](#) section 2.2.2.15) that specifies the coordinates of the reference point used to position the string. The reference point is defined by the last [EMR_SETTEXTALIGN record \(section 2.3.11.25\)](#). If no such record has been set, the default alignment is TA_LEFT,TA_TOP.

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 specified in the **Rectangle** field. This field can be a combination of more than one [ExtTextOutOptions](#) enumeration (section [2.1.11](#)) values.

Rectangle (16 bytes): An optional WMF [RectL](#) object ([\[MS-WMF\]](#) section 2.2.2.19) that defines a clipping and/or opaquing rectangle in logical units. This rectangle is applied to the text output performed by the containing record.[<56>](#)

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.

StringBuffer (variable): The character string buffer.

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
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): An array of characters that specify the string to output. The location of this field is specified by the value of **offString** in bytes from the start of this record. The number of characters is specified by the value of **Chars**.

DxBuffer (variable): The optional character spacing buffer.

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
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 unsigned integers that specify the output spacing between the origins of adjacent character cells in logical units. The location of this field is specified by the value of **offDx** in bytes from the start of this record. If

spacing is defined, this field contains the same number of values as characters in the output string.

If the Options field of the EmrText object contains the ETO_PDY flag, then this buffer contains twice as many values as there are characters in the output string, one horizontal and one vertical offset for each, in that order.

If ETO_RTLREADING is specified, characters are laid right to left instead of left to right. No other options affect the interpretation of this field.

The size and encoding of the characters in the **OutputString** is determined by the type of record that contains the EmrText object, as follows:

- [EMR_EXTTEXTOUTA \(section 2.3.5.7\)](#) and [EMR_POLYTEXTOUTA \(section 2.3.5.32\)](#) records: 8-bit ASCII characters.
- [EMR_EXTTEXTOUTW](#) and [EMR_POLYTEXTOUTW](#) records: 16-bit Unicode UTF16-LE characters.

2.2.6 EpsData Object

The EpsData object is a container for EPS data.

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
SizeData																																		
Version																																		
Points																																		
...																																		
...																																		
...																																		
PostScriptData (variable)																																		
...																																		

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

Version (4 bytes): A 32-bit unsigned integer that specifies the PostScript language level. This value MUST be 0x00000001.

Points (24 bytes): An array of three [Point28_4 objects \(section 2.2.23\)](#) that defines the coordinates of the output parallelogram using 28.4 bit FIX notation.

The upper-left corner of the parallelogram is the first point in this array, the upper-right corner is the second point, and the lower-left corner is the third point. The lower-right corner of the parallelogram is computed from the first three points (A, B, and C) by treating them as vectors.

$$D = B + C - A$$

PostScriptData (variable): An array of bytes of PostScript data. The length of this array can be computed from the **SizeData** field. This data MAY be used to render an image. [\(<57>\)](#)

An EpsData object can be used to embed a PostScript image in an EMF metafile as follows:

- An EMF metafile contains an [EMR_COMMENT_MULTIFORMATS record \(section 2.3.3.4.3\)](#).
- The EMR_COMMENT_MULTIFORMATS record specifies an **aFormats** field that contains an [EmrFormat object \(section 2.2.4\)](#).
- The EmrFormat object specifies a **Signature** field that is set to **EPS_SIGNATURE** from the [FormatSignature enumeration \(section 2.1.14\)](#).
- The **EPS_SIGNATURE** value specifies that the **FormatData** field in the EMR_COMMENT_MULTIFORMATS record contains an EpsData object.
- The EmrFormat object also specifies an **offData** field that indicates where the EpsData object is in the **FormatData** field in the EMR_COMMENT_MULTIFORMATS record.

2.2.7 GradientRectangle Object

The GradientRectangle object defines a rectangle using [TriVertex objects \(section 2.2.26\)](#) in an [EMR_GRADIENTFILL record \(section 2.3.5.12\)](#).

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
UpperLeft																																		
LowerRight																																		

UpperLeft (4 bytes): An index into an array of TriVertex objects that 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 that 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.

2.2.8 GradientTriangle Object

The GradientTriangle object defines a triangle using [TriVertex objects \(section 2.2.26\)](#) in an [EMR_GRADIENTFILL record \(section 2.3.5.12\)](#).

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
Vertex1																																		
Vertex2																																		
Vertex3																																		

Vertex1 (4 bytes): An index into an array of TriVertex objects that 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 that 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 that 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.

2.2.9 Header Object

The Header object defines the EMF metafile header. It specifies properties of the device on which the image in the metafile was created.

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
Bounds																																		
...																																		
...																																		
...																																		
Frame																																		
...																																		
...																																		
RecordSignature																																		
Version																																		

Bytes	
Records	
Handles	Reserved
nDescription	
offDescription	
nPalEntries	
Device	
...	
Millimeters	
...	

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) that 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 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.

RecordSignature (4 bytes): A 32-bit unsigned integer that specifies the record signature. This MUST be **ENHMETA_SIGNATURE**, from the [FormatSignature enumeration \(section 2.1.14\)](#).

Version (4 bytes): A 32-bit unsigned integer that specifies EMF metafile interoperability. This SHOULD be 0x00010000. [<58>](#)

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 graphics objects that will be used during the processing of the metafile.

Reserved (2 bytes): A 16-bit unsigned integer that MUST be 0x0000 and MUST be ignored.

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 WMF [SizeL](#) object ([\[MS-WMF\]](#) section 2.2.2.22) that specifies the size of the reference device, in pixels.

Millimeters (8 bytes): A WMF [SizeL](#) object that specifies the size of the reference device, in millimeters.

2.2.10 HeaderExtension1 Object

The HeaderExtension1 object defines the first extension to the EMF metafile header. It adds support for a [PixelFormatDescriptor](#) object (section [2.2.22](#)) and OpenGL [\[OPENGL\]](#) records (section [2.3.9](#)).

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
cbPixelFormat																																		
offPixelFormat																																		
bOpenGL																																		

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

offPixelFormat (4 bytes): A 32-bit unsigned integer that specifies the offset to the [PixelFormatDescriptor](#) object. 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
0x00000000	OpenGL records are not present in the metafile.
0x00000001	OpenGL records are present in the metafile.

2.2.11 HeaderExtension2 Object

The HeaderExtension2 object defines the second extension to the EMF metafile header. It adds the ability to measure device surfaces in micrometers, which enhances the resolution and scalability of EMF metafiles.

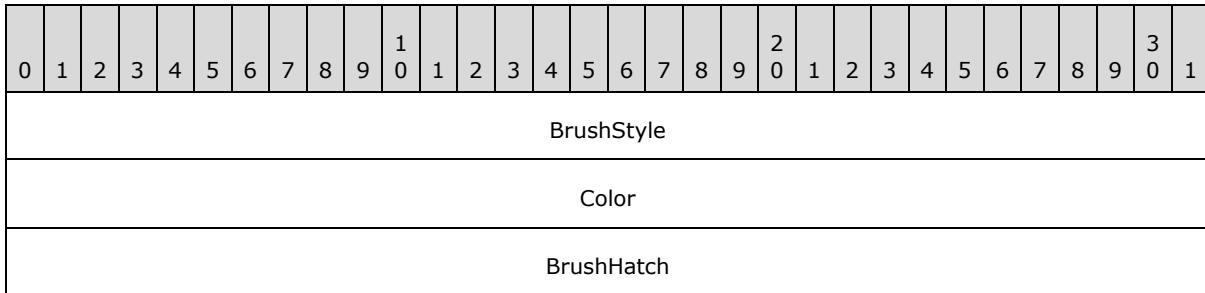
0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
MicrometersX																																		
MicrometersY																																		

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.

2.2.12 LogBrushEx Object

The LogBrushEx object defines the style, color, and pattern of a device-independent brush.



BrushStyle (4 bytes): A 32-bit unsigned integer that specifies the brush style. The value MUST be an enumeration from WMF **BrushStyle** enumeration ([\[MS-WMF\]](#) section 2.1.1.4). 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. [<59>](#)

Color (4 bytes): A 32-bit WMF **ColorRef** object ([\[MS-WMF\]](#) section 2.2.2.8) that specifies a color. The interpretation of this field depends on the value of **BrushStyle**, as explained in the following table.

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

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
BS_SOLID	SHOULD be a WMF ColorRef object, which specifies the color of the brush.	Not used, and SHOULD be ignored.
BS_NULL	Not used, and SHOULD be ignored.	Not used, and SHOULD be ignored.
BS_HATCHED	SHOULD be a WMF ColorRef object, which specifies the foreground color of the hatch pattern.	SHOULD be a value from the EMF HatchStyle (section 2.1.17) enumeration, which specifies the orientation of lines used to create the hatch.

2.2.13 LogFont Object

The LogFont object specifies the basic attributes of a logical font.

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																									
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 size**, is the character **cell height** value minus the **internal leading** value. The font mapper SHOULD interpret the value specified in the **Height** field in the following manner.

Value	Meaning
0x00000000 < value	The font mapper transforms this value into device units and matches it against the cell height of the available fonts.
0x00000000	The font mapper uses a default height value when it searches for a match.
value <	The font mapper transforms this value into device units and matches its

Value	Meaning
0x00000000	absolute value against the character height of the available fonts.

For all height comparisons, the font mapper SHOULD look 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 the **Width** field value is zero, an appropriate value SHOULD be calculated from other LogFont values to find a font that has the typographer's intended **aspect ratio.** [<60>](#)

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 **baseline** 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 baseline 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 zero through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight can be used. [<61>](#)

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

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

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

CharSet (1 byte): An 8-bit unsigned integer that specifies the set of character glyphs. It MUST be a value in the WMF **CharacterSet** enumeration ([\[MS-WMF\]](#) section 2.1.1.5). If the character set is unknown, metafile processing SHOULD NOT attempt to translate or interpret strings that are rendered with that font.

If a typeface name is specified in the **Facename** field, the **CharSet** field value MUST match the character set of that typeface.

OutPrecision (1 byte): An 8-bit unsigned integer that specifies the output precision. The output precision defines how closely the font is required to match the requested height, width, character orientation, escapement, **pitch**, and font type. It MUST be a value from the WMF **OutPrecision** enumeration ([\[MS-WMF\]](#) section 2.1.1.21).

Applications can use the output precision 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 **rasterized** and TrueType forms, an output precision value of **OUT_TT_PRECIS** forces the font mapper to choose the TrueType version. A value of **OUT_TT_ONLY_PRECIS** forces the font mapper to choose a TrueType font, even if it is necessary to substitute a TrueType font with another name.

ClipPrecision (1 byte): An 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 WMF [ClipPrecision Flags](#) ([MS-WMF] section 2.1.2.1).

Quality (1 byte): An 8-bit unsigned integer that specifies the output quality. The output quality defines how closely 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 ([\[MS-WMF\]](#) section 2.1.1.10).

PitchAndFamily (1 byte): A WMF [PitchAndFamily object](#) ([MS-WMF](#) section 2.2.2.14) that specifies the pitch and family of the font. Font families describe the look of a font in a general way. They are intended for specifying a font when the specified typeface 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.

2.2.14 LogFontEx Object

The `LogFontEx` object specifies the extended attributes of a logical font.

...
...
...
...
...
(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 \(section 2.2.13\)](#) object that specifies the basic attributes of the logical font.

FullName (128 bytes): A string of 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 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.

Script (64 bytes): A string of 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.

2.2.15 LogFontExDv Object

The LogFontExDv object specifies the design vector for an extended logical font.

LogFontEx (348 bytes): A [LogFontEx object \(section 2.2.14\)](#) that specifies the extended attributes of the logical font.

DesignVector (variable): A [DesignVector object \(section 2.2.3\)](#). This field MUST NOT be longer than 72 bytes.

A design vector **SHOULD** be specified only for a multiple master OpenType font.

2.2.16 LogFontPanose Object

The LogFontPanose object specifies the PANOSE characteristics of a logical font.

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
LogFont																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		

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

LogFont (92 bytes): A [LogFont \(section 2.2.13\)](#) object that specifies the basic attributes of the logical font.

FullName (128 bytes): A string of 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 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): This field MUST be ignored.

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

Match (4 bytes): This field MUST be ignored.

Reserved (4 bytes): A 32-bit unsigned integer that MUST be set to zero and MUST be ignored.

VendorId (4 bytes): This field MUST be ignored.

Culture (4 bytes): A 32-bit unsigned integer that MUST be set to zero and MUST be ignored.

Panose (10 bytes): A [Panose object \(section 2.2.21\)](#) that specifies the PANOSE characteristics of the logical font. If all fields of this object are zero, it MUST be ignored.

Padding (2 bytes): A field that exists only to ensure 32-bit alignment of this structure. It MUST be ignored.

2.2.17 LogPalette Object

The LogPalette object specifies a logical_palette that contains device-independent color definitions.

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
Version																NumberOfEntries																		
PaletteEntries (variable)																																		
...																																		

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

NumberOfEntries (2 bytes): A 16-bit unsigned integer that specifies the number of [LogPaletteEntry](#) objects (section 2.2.18) in the **PaletteEntries** field.

PaletteEntries (variable): An array of LogPaletteEntry objects that defines 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.

2.2.18 LogPaletteEntry Object

The LogPaletteEntry object defines the values that make up a single entry in a [LogPalette](#) object (section 2.2.17).

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
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.

2.2.19 LogPen Object

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

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
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 WMF **PointL** object ([\[MS-WMF\]](#) section 2.2.2.15) that specifies the width of the pen by the value of its **x** field. The value of its **y** field MUST be ignored.

If the pen type in the **PenStyle** field is **PS_GEOMETRIC**, this value is the width in logical units; otherwise, the width is specified in device units. If the pen type in the **PenStyle** field is **PS_COSMETIC**, this value MUST be 0x00000001.

ColorRef (4 bytes): A WMF **ColorRef** object ([\[MS-WMF\]](#) section 2.2.2.8) that specifies the pen color value.

2.2.20 LogPenEx Object

The LogPenEx object specifies the style, width, and color of an extended logical pen.

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
PenStyle																																		
Width																																		
BrushStyle																																		

ColorRef
BrushHatch
NumStyleEntries
StyleEntry (variable)
...

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

A pen style is a combination of pen type, line style, line cap, and line join.

Width (4 bytes): A 32-bit unsigned integer that specifies the width of the line drawn by the pen.

If the pen type in the **PenStyle** field is **PS_GEOMETRIC**, this value is the width in logical units; otherwise, the width is specified in device units. If the pen type in the **PenStyle** field is **PS_COSMETIC**, this value MUST be 0x00000001.

BrushStyle (4 bytes): A 32-bit unsigned integer that specifies a brush style for the pen from the WMF **BrushStyle** enumeration ([\[MS-WMF\]](#) section 2.1.1.4).

If the pen type in the **PenStyle** field is **PS_GEOMETRIC**, this value MUST be either **BS_SOLID** or **BS_HATCHED**. The value of this field can be **BS_NULL**, but only if the line style specified in **PenStyle** is **PS_NULL**. The **BS_NULL** style SHOULD be used to specify a brush that has no effect.[<62>](#)

ColorRef (4 bytes): A WMF **ColorRef** object ([\[MS-WMF\]](#) section 2.2.2.8). The interpretation of this field depends on the **BrushStyle** value, as shown in the table later in this section.

BrushHatch (4 bytes): The brush hatch pattern. The definition of this field depends on the **BrushStyle** value, as shown in the table later in this section.

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): An optional array of 32-bit unsigned integers that defines the lengths of dashes and gaps in the line drawn by this pen, when the value of **PenStyle** is **PS_USERSTYLE** line style for the pen. The array contains a number of entries specified by **NumStyleEntries**, 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.

If the pen type in the **PenStyle** field is **PS_GEOMETRIC**, the lengths are specified in logical units; otherwise, the lengths are specified in device units.

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

BrushStyle	ColorRef	BrushHatch
BS_SOLID	SHOULD be a WMF ColorRef (section 2.2.2.8) object ([MS-WMF]), which specifies the color of lines drawn by the pen.	Not used, and SHOULD be ignored.
BS_NULL	Not used, and SHOULD be ignored.	Not used, and SHOULD be ignored.
BS_HATCHED	SHOULD be a WMF ColorRef (section 2.2.2.8) object, which specifies the foreground color of the hatch pattern.	SHOULD be a value from the EMF HatchStyle (section 2.1.17) enumeration that specifies the orientation of lines used to create the hatch. If PS_GEOEMTRIC is not set in the PenStyle field, this field MUST be either HS_SOLIDTEXTCLR (0x0008) or HS_SOLIDBKCLR (0x000A) .
BS_PATTERN	The low-order word SHOULD be a value from the WMF ColorUsage (section 2.1.16) enumeration ([MS-WMF] section 2.1.1.6).	Not used, and SHOULD be ignored. The brush pattern is specified by a packed DIB ([MS-WMF] section 2.2.2.9) .
BS_DIBPATTERN	The low-order word SHOULD be a value from the WMF ColorUsage enumeration.	Not used, and SHOULD be ignored. The brush pattern is specified by a packed DIB.
BS_DIBPATTERNPT	The low-order word SHOULD be a value from the WMF ColorUsage (section 2.1.1.6) enumeration.	Not used, and SHOULD be ignored. The brush pattern is specified by a packed DIB.

2.2.21 Panose Object

The 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	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
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 \(section 2.1.34\)](#) 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 \(section 2.1.35\)](#) enumeration table.

2.2.22 PixelFormatDescriptor Object

The PixelFormatDescriptor object specifies the pixel format of a drawing surface.

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																						
nSize																nVersion																																								
dwFlags																																																								
iPixelType								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 that is used for output to the drawing surface. These properties are not all mutually exclusive; combinations of flags are allowed, except where noted otherwise.

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
P	F	S	G	M	W	S	D	C	D	A	S	A	S	S	S	S	P	0	0	0	0	0	0	0	0	0	0	0	0	S	D	D	P	0

The following bit flag constants are defined.

Value	Description
D PFD_DOUBLEBUFFER	The pixel buffer is double-buffered. This flag and PFD_SUPPORT_GDI MUST NOT both be set.
S PFD_STEREO	The pixel buffer MAY be stereoscopic ; that is, it MAY specify a color plane that is used to create the illusion of depth in an image. <63>
W PFD_DRAW_TO_WINDOW	The pixel buffer can draw to a window or device surface.
M PFD_DRAW_TO_BITMAP	The pixel buffer can draw to a memory bitmap.
G PFD_SUPPORT_GDI	This flag SHOULD be clear, but it MAY be set. <64> The PFD_SUPPORT_GDI flag and PFD_DOUBLEBUFFER MUST NOT both be set.
SO PFD_SUPPORT_OPENGL	The pixel buffer supports OpenGL drawing. See [OPENGL] for more information.
F PFD_GENERIC_FORMAT	The pixel format is natively supported by the operating system; this is known as the "generic" implementation. <65> If clear, the pixel format is supported by a device driver or hardware.
P PFD_NEED_PALETTE	The buffer uses RGBA pixels on a palette-managed device. A LogPalette object (section 2.2.17) is required to achieve the best results for this pixel type. Colors in the palette SHOULD be specified according to the values of the cRedBits , cRedShift , cGreenBits , cGreenShift , cBlueBits , and cBlueShift fields.
SP PFD_NEED_SYSTEM_PALETTE	The output device 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 LogPalette object when in color table mode.
SE	The contents of the back buffer have been exchanged with the

Value	Description
PFD_SWAP_EXCHANGE	contents of the front buffer in a double-buffered color plane.
SC PFD_SWAP_COPY	The contents of the back buffer have been copied to the front buffer in a double-buffered color plane. The contents of the back buffer have not been affected.
SL PFD_SWAP_LAYER_BUFFERS	A device can swap individual color planes with pixel formats that include double-buffered overlay or underlay color planes. Otherwise all color planes are swapped together as a group.
A PFD_GENERIC_ACCELERATED	The pixel format is supported by a device driver that accelerates the generic implementation. If this flag is clear and the PFD_GENERIC_FORMAT flag is set, the pixel format is supported by the generic implementation only.
DS PFD_SUPPORT_DIRECTDRAW	The pixel buffer supports DirectDraw drawing, which allows applications to have low-level control of the output drawing surface.
DA PFD_DIRECT3D_ACCELERATED	The pixel buffer supports Direct3D drawing, which accelerates rendering in three dimensions.
C PFD_SUPPORT_COMPOSITION	The pixel buffer supports compositing , which indicates that source pixels MAY overwrite or be combined with background pixels. <66>
DP PFD_DEPTH_DONTCARE	The pixel buffer is not required to include a color plane for depth effects.
DD PFD_DOUBLEBUFFER_DONTCARE	The pixel buffer can be either single or double buffered.
SD PFD_STEREO_DONTCARE	The pixel buffer MAY be either monoscopic or stereoscopic.

iPixelType (1 byte): The type of pixel data.

Value	Meaning
PFD_TYPE_RGBA 0x00	The pixel format is RGBA.
PFD_TYPE_COLORINDEX 0x01	Each pixel is an index in a color table.

cColorBits (1 byte): The number of bits per pixel for RGBA pixel types, excluding the alpha bitplanes. For color table pixels, it is the size of each color table index.

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.[<67>](#)

cAlphaShift (1 byte): Specifies the shift count for alpha bitplanes in each RGBA color buffer.[<68>](#)

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.[<69>](#)

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.

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.

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.

The PixelFormatDescriptor object can be used in [EMR_HEADER records \(section 2.3.4.2\)](#) to specify the pixel format of the output surface for the playback device context.

2.2.23 Point28_4 Object

The Point28_4 object represents the location of a point on a device surface with coordinates in 28.4 bit FIX notation.

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
x																																		
y																																		

x (4 bytes): A [BitFIX28_4 object \(section 2.2.1\)](#) that represents the horizontal coordinate of the point.

y (4 bytes): A BitFIX28_4 object that represents the vertical coordinate of the point.

2.2.24 RegionData Object

The 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	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
RegionDataHeader																																		
...																																		
...																																		
...																																		
...																																		
...																																		
Data (variable)																																		
...																																		

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

Data (variable): An array of WMF [RectL](#) objects ([\[MS-WMF\]](#) section 2.2.2.19); the objects are merged to create the region.

2.2.25 RegionDataHeader Object

The RegionDataHeader object describes the properties of a [RegionData](#) object.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19), which specifies the bounds of the region.

2.2.26 TriVertex Object

The 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	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
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.

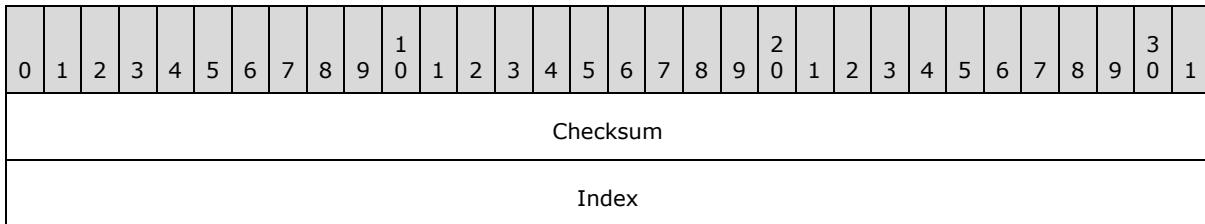
Green (2 bytes): A 16-bit unsigned integer that specifies the green color value for the point.

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.

2.2.27 UniversalFontId Object

The UniversalFontId object defines a mechanism for identifying fonts in EMF metafiles.



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 Type 1 font that has been installed on the client machine and is enumerated by the PostScript printer driver as a device font.
0x00000002	The object is not a font but is a Type 1 rasterizer .
$3 \leq value$	The object is a bitmap, vector , or TrueType font, or a Type 1 rasterized font that was created by a Type 1 rasterizer. A checksum value SHOULD be computed for the font and compared to the value in this field. If it matches, it is considered to be the same as the font referenced by this metafile record. If it does not match, the system font mapper MAY use a default mechanism to select a back-up font. <70>

If a checksum value is computed, it SHOULD be computed using the following algorithm.

Note For the purpose of this computation, the font is considered simply to be a stream of bytes that is external to this EMF record. Any larger file structure in which the font might reside is system-dependent or implementation-dependent.

```
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.

cjView: The length of the font 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.

2.2.28 XForm Object

The XForm object defines a two-dimensional, linear transform matrix.

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
M11																																		
M12																																		
M21																																		
M22																																		
Dx																																		
Dy																																		

M11 (4 bytes): A 32-bit floating-point value of the transform matrix.

M12 (4 bytes): A 32-bit floating-point value of the transform matrix.

M21 (4 bytes): A 32-bit floating-point value of the transform matrix.

M22 (4 bytes): A 32-bit floating-point value of the transform matrix.

Dx (4 bytes): A 32-bit floating-point value that contains a horizontal translation component, in logical units.

Dy (4 bytes): A 32-bit floating-point value that contains a vertical translation component, in logical units.

The following equations specify how the matrix values are used to transform a point (X,Y) to a new point (X',Y'):

$$\begin{aligned} X' &= M11 * X + M21 * Y + Dx \\ Y' &= M12 * X + M22 * Y + Dy \end{aligned}$$

For more information concerning transforms and coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

2.3 EMF Records

This section specifies the types of EMF metafile records, which have been grouped into the following categories.

Name	Section	Description
Bitmap record types	2.3.1	Manage and output bitmap images.
Clipping record types	2.3.2	Specify and manage clipping regions.
Comment record types	2.3.3	Define formats for specifying arbitrary private data, embedding records in other metafile formats, and adding new or special-purpose commands.
Control record types	2.3.4	Define the start and end of an EMF metafile and its properties.
Drawing record types	2.3.5	Perform graphics drawing.
Escape record types	2.3.6	Execute printer driver functions.
Object creation record types	2.3.7	Create graphics objects.
Object manipulation record types	2.3.8	Manage and modify graphics objects.
OpenGL record types	2.3.9	Specify metafile records generated by OpenGL.
Path bracket record types	2.3.10	Specify and manipulate paths in path brackets .
State record types	2.3.11	Specify and manage graphics properties.
Transform record types	2.3.12	Specify and modify world-space to page-space transforms.

Note All EMF records MUST have a length that is a multiple of 4 bytes. This is depicted in the generic structures of the preceding EMF record types by including **AlignmentPadding** fields where appropriate at the ends of these structures. The contents of **AlignmentPadding** fields MUST always be ignored. For brevity, these fields are not shown in every individual EMF record definition.

2.3.1 Bitmap Record Types

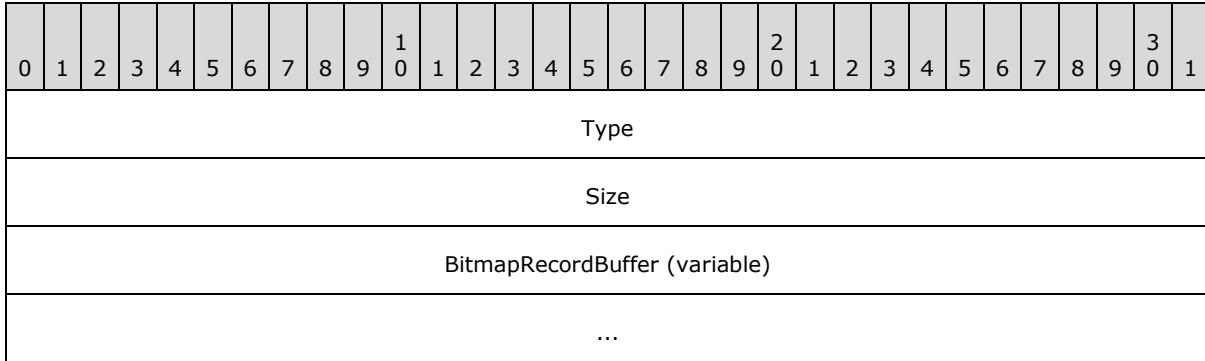
The bitmap record types perform block transfers of bitmap images.

The following are EMF bitmap record types.

Name	Section	Description
EMR_ALPHABLEND	2.3.1.1	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, including alpha transparency data, according to a specified blending operation.
EMR_BITBLT	2.3.1.2	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation.

Name	Section	Description
EMR_MASKBLT	2.3.1.3	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern and with the application of a color mask bitmap, according to specified foreground and background raster operations.
EMR_PLGBLT	2.3.1.4	Specifies a block transfer of pixels from a source bitmap to a destination parallelogram, with the application of a color mask bitmap.
EMR_SETDBITSTODEVICE	2.3.1.5	Specifies a block transfer of pixels from specified scanlines of a source bitmap to a destination rectangle.
EMR_STRETCHBLT	2.3.1.6	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.
EMR_STRETCHDIBITS	2.3.1.7	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.
EMR_TRANSPARENTBLT	2.3.1.8	Specifies a block transfer of pixels from a source bitmap to a destination rectangle, treating a specified color as transparent, stretching or compressing the output to fit the dimensions of the destination, if necessary.

The generic structure of bitmap records is specified as follows.



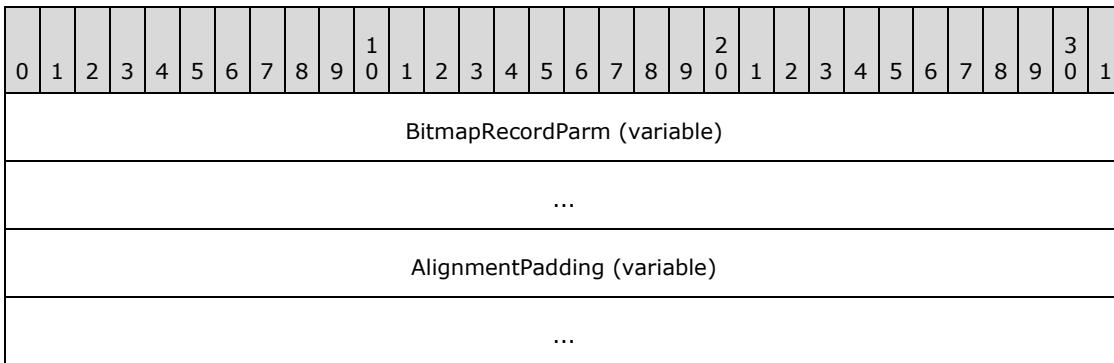
Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The bitmap record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_BITBLT	0x0000004C
EMR_STRETCHBLT	0x0000004D

Name	Value
EMR_MASKBLT	0x0000004E
EMR_PLGBLT	0x0000004F
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 this record in the metafile. This value MUST be a multiple of 4 bytes.

BitmapRecordBuffer (variable): An array of bytes that contains the remainder of the bitmap record. The size of this field MUST be a multiple of 4 bytes.



BitmapRecordParm (variable): An array of bytes that contains the parameters for the bitmap record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

The following notes apply generally to EMF bitmap block transfers, unless specified otherwise:

- Source and mask bitmaps are in DIB format. DIBs are specified by [DeviceIndependentBitmap](#) objects ([\[MS-WMF\]](#) section 2.2.2.9).
- The properties that describe the structure of the destination of the block transfer are defined in the playback device context.
- If the color format of the source or pattern bitmap does not match the color format of the destination, the source or pattern bits MUST be converted to the destination format prior to performing the block transfer.
- If the source and destination rectangles are not the same size, the source bitmap MUST be expanded or compressed to match the destination rectangle. This stretching function is performed according to a property in the playback device context, which is from the [StretchMode](#) enumeration (section [2.1.32](#)).

- If an [XForm](#) object (section [2.2.28](#)) is specified, a world-space to page-space transform SHOULD be applied to the source bitmap. Scaling, translation, and **reflection transforms** SHOULD be supported, and rotation and shear transforms MAY be supported.[<71>](#)

For more information concerning transforms and coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

See section [2.3](#) for additional EMF record types.

2.3.1.1 EMR_ALPHABLEND Record

The **EMR_ALPHABLEND** record specifies a block transfer of pixels from a source bitmap to a destination rectangle, including alpha transparency data, according to a specified blending operation.[<72>](#)

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
cxDest																																		
cyDest																																		
BLENDFUNCTION																																		
xSrc																																		
ySrc																																		
XformSrc																																		
...																																		

...
...
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_ALPHABLEND**. This MUST be 0x00000072.

Bounds (16 bytes): A WMF [RectL object](#) ([MS-WMF] section 2.2.2.19) that defines the destination 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 zero.

cyDest (4 bytes): A 32-bit signed integer that specifies the logical height of the destination rectangle. This value MUST be greater than zero.

BLENDFUNCTION (4 bytes): A structure that specifies the blending operations for source and destination bitmaps.

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
BlendOperation								BlendFlags								SrcConstantAlpha								AlphaFormat										

BlendOperation (1 byte): The blend operation code. The only source and destination blend operation that has been defined is 0x00, 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 following equations for details.

BlendFlags (1 byte): This value MUST be 0x00 and MUST be ignored.

SrcConstantAlpha (1 byte): An 8-bit unsigned integer that specifies alpha transparency, which determines the blend of the source and destination bitmaps. This value MUST be used on the entire source bitmap. The minimum alpha transparency value, zero, corresponds to completely transparent; 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 later in this section for details.

AlphaFormat (1 byte): A structure that specifies how source and destination pixels are interpreted with respect to alpha transparency.

Value	Meaning
0x00	The pixels in the source bitmap do not specify alpha transparency. In this case, the SrcConstantAlpha value determines the blend of the source and destination bitmaps. Note that in the following equations SrcConstantAlpha is divided by 255, which produces a value in the range 0 to 1.
AC_SRC_ALPHA 0x01	Indicates that the source bitmap is 32 bits-per-pixel and specifies 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): An [XForm](#) object (section [2.2.28](#)) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8 that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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 in the **BitmapBuffer** field.

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 in the **BitmapBuffer** field.

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 zero.

cySrc (4 bytes): A 32-bit signed integer that specifies the logical height of the source rectangle. This value MUST be greater than zero.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the **EMR_ALPHABLEND** record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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	1
UndefinedSpace1 (variable)																																	
...																																	
BmiSrc (variable)																																	
...																																	
UndefinedSpace2 (variable)																																	
...																																	
BitsSrc (variable)																																	
...																																	

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

The following equations 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 premultiplied 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, 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 If **SrcConstantAlpha** is 0xFF, there is in effect no premultiplication of the source values.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.2 EMR_BITBLT Record

The EMR_BITBLT record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
cxDest																																		
cyDest																																		
BitBltRasterOperation																																		
xSrc																																		
ySrc																																		
XformSrc																																		
...																																		
...																																		
...																																		
BkColorSrc																																		
UsageSrc																																		
offBmiSrc																																		

cbBmiSrc
offBitsSrc
cbBitsSrc
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as EMR_BITBLT. This MUST be 0x0000004C.

Bounds (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that defines the destination 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 ([\[MS-WMF\]](#) section 2.1.1.31).

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): An [XForm object](#) (section 2.2.28) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF [ColorRef object](#) ([\[MS-WMF\]](#) section 2.2.2.8) that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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 in the **BitmapBuffer** field.

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 in the **BitmapBuffer** field.

cbBitsSrc (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the source bitmap bits.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the EMR_BITBLT record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note If the raster operation specified by **BitBltRasterOperation** does not require a source bitmap, the source bitmap can be omitted.

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	1
UndefinedSpace1 (variable)																																	
...																																	
BmiSrc (variable)																																	
...																																	
UndefinedSpace2 (variable)																																	
...																																	
BitsSrc (variable)																																	
...																																	

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.3 EMR_MASKBLT Record

The EMR_MASKBLT record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern and with the application of a color mask bitmap, according to specified foreground and background raster operations.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
cxDest																																		
cyDest																																		
ROP4																																		
xSrc																																		
ySrc																																		
XformSrc																																		
...																																		
...																																		
...																																		
...																																		
BkColorSrc																																		
UsageSrc																																		
offBmiSrc																																		

cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_MASKBLT**.
This MUST be 0x0000004E.

Bounds (16 bytes): A WMF [RectL](#) object ([\[MS-WMF\]](#) section 2.2.2.19) that defines the destination 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. These values define how the color data of the source rectangle is to be combined with the color data of the destination rectangle.

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
Reserved												BackgroundROP3												ForegroundROP3										

Reserved (2 bytes): This field SHOULD be 0x0000 and MUST be ignored.[<73>](#)

BackgroundROP3 (1 byte): The unsigned, most-significant 8 bits of a 24-bit ternary raster operation value from the WMF [Ternary Raster Operation](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.31). 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): An [XForm](#) object (section [2.2.28](#)) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8) that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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 in the **BitmapBuffer** field.

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 in the **BitmapBuffer** field.

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 logical x-coordinate of the upper-left corner of the mask bitmap.

yMask (4 bytes): A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the mask bitmap.

UsageMask (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the mask bitmap header. This 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 in the **BitmapBuffer** field.

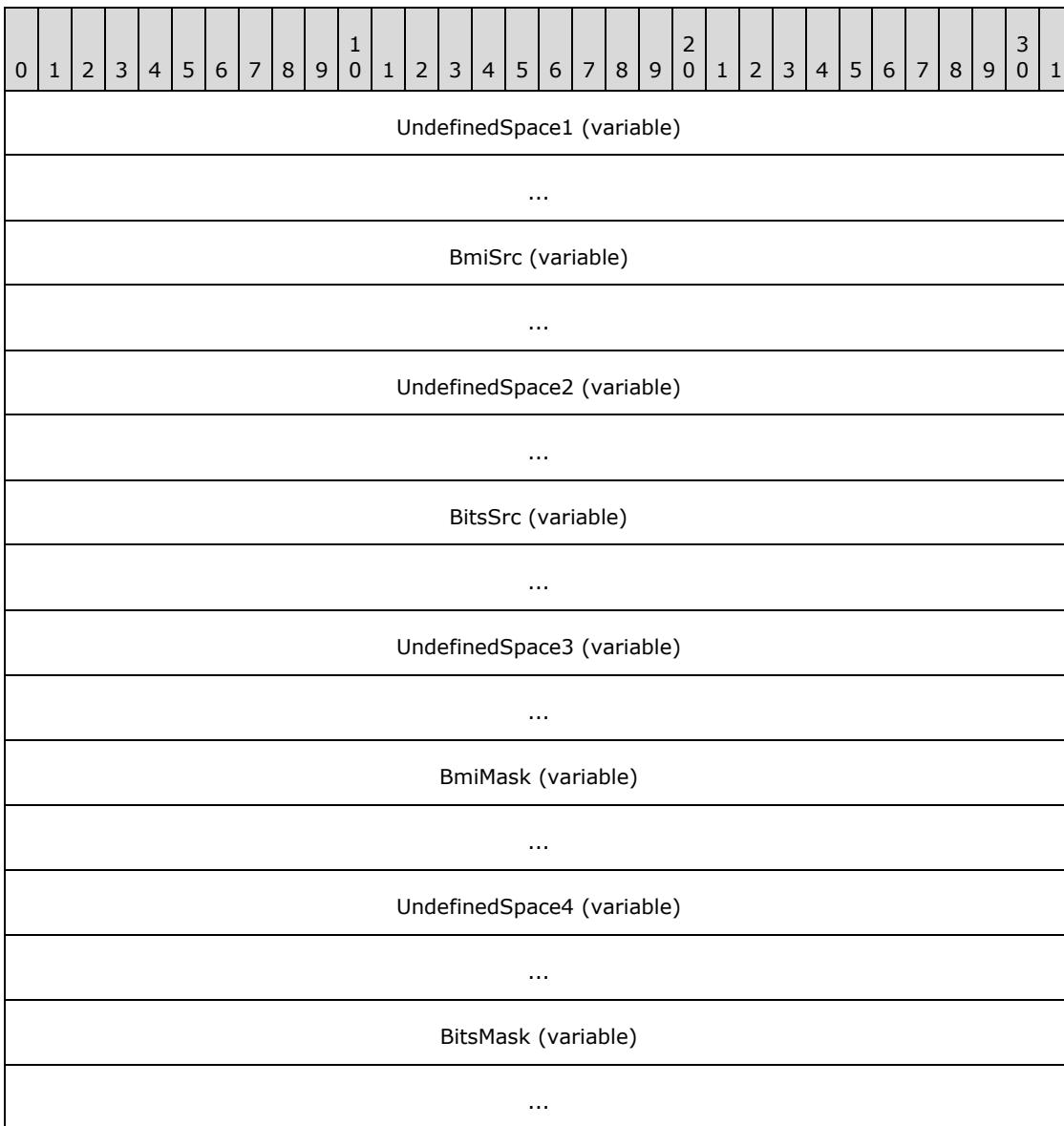
cbBmiMask (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the mask bitmap header.

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 in the **BitmapBuffer** field.

cbBitsMask (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the mask bitmap bits.

BitmapBuffer (variable): A buffer containing the source and mask bitmaps, which are not required to be contiguous with the fixed portion of the EMR_MASKBLT record or with each other. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note The source and mask bitmaps can be present in this buffer in any order.



BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

BmiMask (variable): The mask bitmap header.

BitsMask (variable): The mask bitmap bits.

The mask bitmap MUST be monochrome; that is, each pixel value MUST be zero or one. A pixel value of one in the mask indicates that the color of the corresponding pixel in the source bitmap SHOULD be copied to the destination. A value of zero 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.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.4 EMR_PLGBLT Record

The EMR_PLGBLT record specifies a block transfer of pixels from a source bitmap to a destination parallelogram, with the application of a color mask bitmap.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
aptIDest																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
xSrc																																		
ySrc																																		

cxSrc
cySrc
XformSrc
...
...
...
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_PLGBLT**.
This MUST be 0x0000004F.

Bounds (16 bytes): A WMF [RectL](#) object ([\[MS-WMF\]](#) section 2.2.2.19) that defines the bounding rectangle, in device units, for output to the destination.

aptlDest (24 bytes): An array of three WMF [PointL](#) objects ([\[MS-WMF\]](#) section 2.2.2.15) that specifies three corners a parallelogram destination area for the block transfer.

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, which is computed from the first three points (A, B, and C) by treating them as vectors.

D = B + C A

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): An [XForm](#) object (section [2.2.28](#)) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8) that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the **DIBColors** enumeration (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 in the **BitmapBuffer** field.

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 in the **BitmapBuffer** field.

cbBitsSrc (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the source bitmap.

xMask (4 bytes): A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the mask bitmap.

yMask (4 bytes): A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the mask bitmap.

UsageMask (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the mask bitmap header. This 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 header of the mask bitmap in the **BitmapBuffer** field.

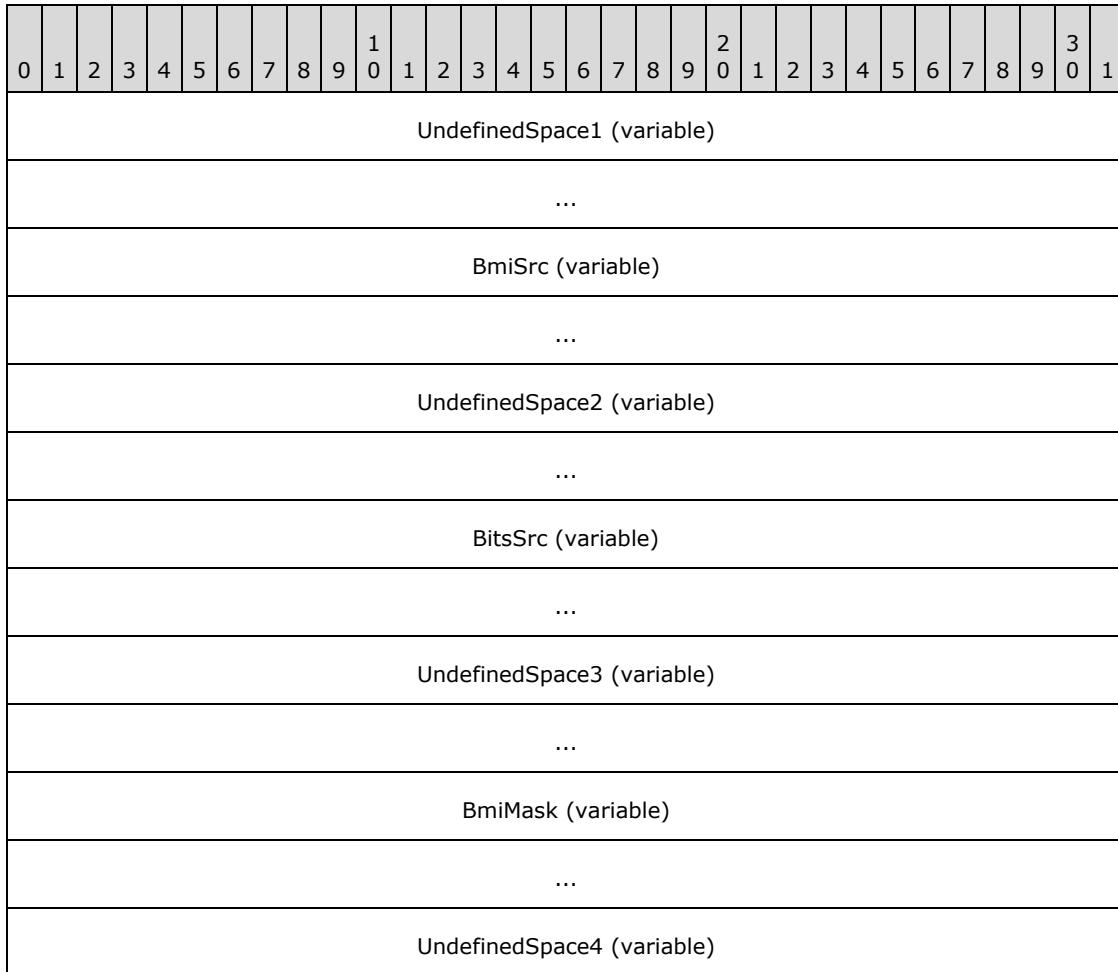
cbBmiMask (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the mask bitmap header.

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 in the **BitmapBuffer** field.

cbBitsMask (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the mask bitmap bits.

BitmapBuffer (variable): A buffer containing the source and mask bitmaps, which are not required to be contiguous with the fixed portion of the EMR_PLGBT record or with each other. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note The source and mask bitmaps can be present in any order.



...
BitsMask (variable)
...

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

BmiMask (variable): The mask bitmap header.

BitsMask (variable): The mask bitmap bits.

The mask bitmap MUST be monochrome; that is, each pixel value MUST be zero or one. A pixel value of one in the mask indicates that the color of the corresponding pixel in the source bitmap SHOULD be copied to the destination. A value of zero 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.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.5 EMR_SETDIBITSTODEVICE Record

The EMR_SETDIBITSTODEVICE record specifies a block transfer of pixels from specified scanlines of a source bitmap to a destination rectangle.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
xSrc																																		

ySrc
cxSrc
cySrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
UsageSrc
iStartScan
cScans
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_SETDIBITSTODEVICE**. This MUST be 0x00000050.

Bounds (16 bytes): A WMF [RectL object](#) ([MS-WMF] section 2.2.2.19) that defines the destination 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 x-coordinate in pixels of the lower-left corner of the source rectangle.

ySrc (4 bytes): A 32-bit signed integer that specifies the y-coordinate in pixels of the lower-left corner of the source rectangle.

cxSrc (4 bytes): A 32-bit signed integer that specifies the width in pixels of the source rectangle.

cySrc (4 bytes): A 32-bit signed integer that specifies the height in pixels 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 in the **BitmapBuffer** field.

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 in the **BitmapBuffer** field.

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 values in the color table in the source bitmap header. This value MUST be in the **DIBColors** enumeration (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.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the EMR_SETDIBITSTODEVICE record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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
UndefinedSpace1 (variable)																																		
...																																		
BmiSrc (variable)																																		
...																																		
UndefinedSpace2 (variable)																																		
...																																		
BitsSrc (variable)																																		
...																																		

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

This record supports source images in **JPEG** and **PNG** format. The **Compression** field in the source bitmap header specifies the image format.

See section [2.3.1.1](#) for additional bitmap record types.

2.3.1.6 EMR_STRETCHBLT Record

The EMR_STRETCHBLT record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster

operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
cxDest																																		
cyDest																																		
BitBltRasterOperation																																		
xSrc																																		
ySrc																																		
XformSrc																																		
...																																		
...																																		
...																																		
...																																		
...																																		
BkColorSrc																																		

UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_STRETCHBLT**. This MUST be 0x0000004D.

Bounds (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that defines the destination 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.

This value MUST be in the WMF **Ternary Raster Operation** enumeration ([\[MS-WMF\]](#) section 2.1.1.31).

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): An [XForm object](#) (section 2.2.28) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8 that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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.

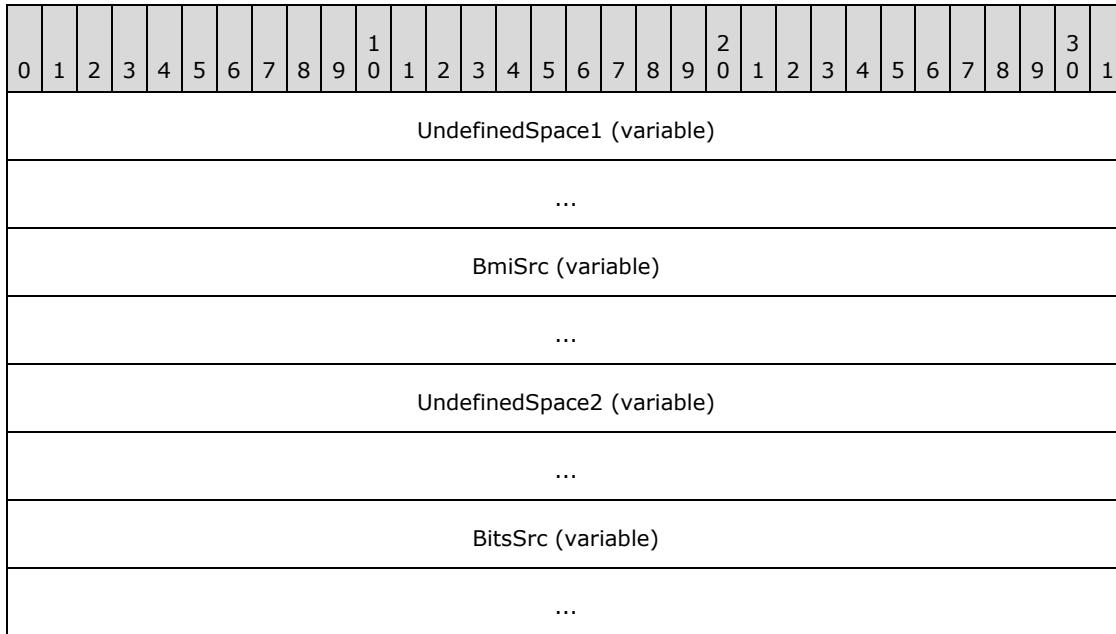
cbBitsSrc (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the source bitmap bits.

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.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the EMR_STRETCHBLT record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note If the raster operation specified by **BitBltRasterOperation** does not require a source bitmap, the source bitmap can be omitted.



BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.7 EMR_STRETCHDIBITS Record

The EMR_STRETCHDIBITS record specifies a block transfer of pixels from a source bitmap to a destination rectangle, optionally in combination with a brush pattern, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
xSrc																																		
ySrc																																		
cxSrc																																		
cySrc																																		
offBmiSrc																																		
cbBmiSrc																																		
offBitsSrc																																		
cbBitsSrc																																		
UsageSrc																																		

BitBltRasterOperation
cxDest
cyDest
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_STRETCHDIBITS**. This MUST be 0x00000051.

Bounds (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that defines the destination 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 x-coordinate in pixels of the upper-left corner of the source rectangle.

ySrc (4 bytes): A 32-bit signed integer that specifies the y-coordinate in pixels of the upper-left corner of the source rectangle.

cxSrc (4 bytes): A 32-bit signed integer that specifies the width in pixels of the source rectangle.

cySrc (4 bytes): A 32-bit signed integer that specifies the height in pixels 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 values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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 ([\[MS-WMF\]](#) section 2.1.1.31).

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.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the EMR_STRETCHDIBITS record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note If the raster operation specified by **BitBltRasterOperation** does not require a source bitmap, the source bitmap can be omitted.

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	1
UndefinedSpace1 (variable)																																	
...																																	
BmiSrc (variable)																																	
...																																	
UndefinedSpace2 (variable)																																	
...																																	
BitsSrc (variable)																																	
...																																	

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

This record supports source images in JPEG and PNG formats. The **Compression** field in the source bitmap header specifies the image format.

If the signs of the source and destination height and width fields differ, this record specifies a mirror-image copy of the source bitmap to the destination. That is, if **cxSrc** and **cxDest** have different signs, a mirror image of the source bitmap along the x-axis is specified. If **cySrc** and **cyDest** have different signs, a mirror image of the source bitmap along the y-axis is specified.

See section [2.3.1](#) for additional bitmap record types.

2.3.1.8 EMR_TRANSPARENTBLT Record

The EMR_TRANSPARENTBLT record specifies a block transfer of pixels from a source bitmap to a destination rectangle, treating a specified color as transparent, stretching or compressing the output to fit the dimensions of the destination, if necessary. [<74>](#)

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
xDest																																		
yDest																																		
cxDest																																		
cyDest																																		
TransparentColor																																		
xSrc																																		
ySrc																																		
XformSrc																																		
...																																		
...																																		
...																																		
...																																		

BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BitmapBuffer (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_TRANSPARENTBLT**. This MUST be 0x00000074.

Bounds (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that defines the destination 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 WMF [ColorRef object](#) ([\[MS-WMF\]](#) section 2.2.2.8) that specifies the color in the source bitmap to be treated as transparent.

xDst (4 bytes): A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

yDst (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): An [XForm object](#) (section 2.2.28) that specifies a world-space to page-space transform to apply to the source bitmap.

BkColorSrc (4 bytes): A WMF ColorRef object that specifies the background color of the source bitmap.

UsageSrc (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value MUST be in the [DIBColors](#) enumeration (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.

BitmapBuffer (variable): A buffer containing the source bitmap, which is not required to be contiguous with the fixed portion of the EMR_TRANSPARENTBLT record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

Note If the source bitmap color format is 32 bits-per-pixel, only the alpha transparency value in each pixel SHOULD be copied to the destination. [<75>](#)

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
UndefinedSpace1 (variable)																																		
...																																		
BmiSrc (variable)																																		
...																																		
UndefinedSpace2 (variable)																																		
...																																		
BitsSrc (variable)																																		
...																																		

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

See section [2.3.1](#) for additional bitmap record types.

2.3.2 Clipping Record Types

The clipping record types specify and manage clipping regions.

Note The EMR_SETMETARGN record does not specify parameters.

The following are EMF clipping record types.

Name	Section	Description
EMR_EXCLUDECLIPRECT	2.3.2.1	Specifies a new clipping region that consists of the existing clipping region minus the specified rectangle.
EMR_EXTSELECTCLIPRGN	2.3.2.2	Combines the specified region with the current clip region using the specified mode.
EMR_INTERSECTCLIPRECT	2.3.2.3	Specifies a new clipping region from the intersection of the current clipping region and the specified rectangle.
EMR_OFFSETCLIPRGN	2.3.2.4	Specifies the clipping region with the specified offsets.
EMR_SELECTCLIPPATH	2.3.2.5	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 clipping records is specified as follows.

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
Type																																		
Size																																		
ClippingRecordBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The clipping record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_OFFSETCLIPRGN	0x0000001A
EMR_SETMETARGN	0x0000001C
EMR_EXCLUDECLIPRECT	0x0000001D

Name	Value
EMR_INTERSECTCLIPRECT	0x00000001E
EMR_SELECTCLIPPATH	0x000000043
EMR_EXTSELECTCLIPRGN	0x00000004B

Size (4 bytes): A 32-bit unsigned integer that specifies the size in bytes of this record in the metafile. This value MUST be a multiple of 4 bytes.

ClippingRecordBuffer (variable): An optional array of bytes that contains the remainder of the clipping record. The size of this field MUST be a multiple of 4 bytes.

Note The EMR_SETMETARGN record does not contain this field.

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
ClippingRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

ClippingRecordParm (variable): An optional array of bytes that contains the parameters for the clipping record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.2.1 EMR_EXCLUDECLIPRECT Record

The EMR_EXCLUDECLIPRECT record specifies a new clipping region that consists of the existing clipping region minus the specified rectangle.

Note Fields that are not described in this section are specified in section [2.3.2](#).

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
Type																																		
Size																																		
Clip																																		

...
...
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_EXCLUDECLIPRECT**. This MUST be 0x0000001D.

Clip (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) that 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 additional clipping record types.

2.3.2.2 EMR_EXTSELECTCLIPRGN Record

The EMR_EXTSELECTCLIPRGN record combines the specified region with the current clip region using the specified mode.

Note Fields that are not described in this section are specified in section [2.3.2](#).

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
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.

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 a **RegionData** object in logical units. If **RegionMode** is **RGN_COPY**, this data can be omitted and the clip region SHOULD be set to the default (NULL) clip region.

See section [2.3.2](#) for additional clipping record types.

2.3.2.3 EMR_INTERSECTCLIPRECT Record

The EMR_INTERSECTCLIPRECT record specifies a new clipping region from the intersection of the current clipping region and the specified rectangle.

Note Fields that are not described in this section are specified in section [2.3.2](#).

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
Type																																		
Size																																		
Clip																																		
...																																		
...																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_INTERSECTCLIPRECT**. This MUST be 0x0000001E.

Clip (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) that 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 additional clipping record types.

2.3.2.4 EMR_OFFSETCLIPRGN Record

The EMR_OFFSETCLIPRGN record moves the current clipping region in the playback device context by the specified offsets.

Note Fields that are not described in this section are specified in section [2.3.2](#).

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
Type																																		
Size																																		
Offset																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_OFFSETCLIPRGN**. This MUST be 0x0000001A.

Offset (8 bytes): A WMF **PointL** object ([\[MS-WMF\]](#) section 2.2.2.15) that specifies the horizontal and vertical offsets in logical units.

See section [2.3.2](#) for additional clipping record types.

2.3.2.5 EMR_SELECTCLIPPATH Record

The 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.

Note Fields that are not described in this section are specified in section [2.3.2](#).

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
Type																																		
Size																																		
RegionMode																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_SELECTCLIPPATH**. This MUST be 0x00000043.

RegionMode (4 bytes): A 32-bit unsigned integer that specifies the way to use the path. The value MUST be in the [RegionMode](#) enumeration (section [2.1.29](#)).

See section [2.3.2](#) for additional clipping record types.

2.3.3 Comment Record Types

The 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	2.3.3.1	Contains arbitrary private data.
EMR_COMMENT_EMFPLUS	2.3.3.2	Contains embedded EMF+ records.
EMR_COMMENT_EMFSPOOL	2.3.3.3	Contains embedded EMFSPOOL records.
EMR_COMMENT_PUBLIC	2.3.3.4	Specifies extensions to EMF processing.

The generic structure of comment records is specified as follows.

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
Type																																		
Size																																		
DataSize																																		
CommentRecordBuffer (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 in bytes of this record in the metafile. This value MUST be a multiple of 4 bytes.

DataSize (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the **CommentIdentifier** and **CommentRecordParm** fields in the **RecordBuffer** field that follows. It MUST NOT include the size of itself or the size of the **AlignmentPadding** field, if present.

CommentRecordBuffer (variable): An array of bytes that contains the remainder of the comment record. The size of this field MUST be a multiple of 4 bytes.

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
CommentIdentifier (optional)																																		
CommentRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

CommentIdentifier (4 bytes): An optional, 32-bit unsigned integer that identifies the type of comment record. See the preceding table for descriptions of these record types.

Valid comment identifier values are listed in the following table. 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_EMFSPOOL	0x00000000

Name	Value
EMR_COMMENT_EMFPLUS	0x2B464D45
EMR_COMMENT_PUBLIC	0x43494447

CommentRecordParm (variable): An array of bytes that contains the parameters for the comment record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.3.1 EMR_COMMENT Record

The EMR_COMMENT record contains arbitrary private data.

Note Fields that are not described in this section are specified in section [2.3.3](#).

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
Type																																		
Size																																		
DataSize																																		
PrivateData (variable)																																		
...																																		

PrivateData (variable): An optional array of bytes that specifies the private data. The first **DWORD** of this data MUST NOT be one of the predefined comment identifier 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. [<76>](#)

See section [2.3.3](#) for additional comment record types.

2.3.3.2 EMR_COMMENT_EMFPLUS Record

The EMR_COMMENT_EMFPLUS record contains embedded EMF+ records.

Note Fields that are not described in this section are specified in section [2.3.3](#).

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
Type																																		

CommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies this comment record as containing EMF+ records. The value 0x2B464D45, which is the ASCII string "+FME", identifies this as an EMR_COMMENT_EMFPLUS record.

EMFPlusRecords (variable): An array of bytes that contains one or more EMF+ records ([IMS-EMFPLUS](#) section 2.3.1).

See section [2.3.3](#) for additional comment record types.

2.3.3.3 EMR_COMMENT_EMFSPOOL Record

The EMR_COMMENT_EMFSPOOL record contains embedded EMFSPOOL records.

Note Fields that are not described in this section are specified in section [2.3.3](#).

CommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies this comment record as containing EMFSPOOL records. The value 0x00000000 identifies this as an EMR COMMENT EMFSPOOL record.

EMFSpoolRecordIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of EMR_COMMENT_EMFSPOOL record. The following value is defined.

Value	Meaning
EMFSpoolFontDefinition 0x544F4E46	The ASCII string "TONF", which identifies this as a record that contains embedded EMFSPOOL font definition data.

EMFSpoolRecords (variable): A variable-length array of bytes that contains one or more EMFSPOOL font definition records ([\[MS-EMFSPOOL\]](#) section 2.2.3.3).

See section [2.3.3](#) for additional comment record types.

2.3.3.4 EMR_COMMENT_PUBLIC Record Types

The EMR_COMMENT_PUBLIC record types specify extensions to EMF processing.

Following are the EMF public comment record types that have been defined.

Name	Section	Description
EMR_COMMENT_BEGINGROUP	2.3.3.4.1	Specifies the beginning of a group of drawing records.
EMR_COMMENT_ENDDGROUP	2.3.3.4.2	Specifies the end of a group of drawing records.
EMR_COMMENT_MULTIFORMATS	2.3.3.4.3	Specifies an image in multiple graphics formats.
EMR_COMMENT_WINDOW_METAFILE	2.3.3.4.4	Specifies an image in an embedded WMF metafile.

The generic structure of EMR_COMMENT_PUBLIC records is specified as follows.

Note Fields that are not described in this section are specified in section [2.3.3](#).

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
Type																																		
Size																																		
DataSize																																		
CommentIdentifier																																		
PublicCommentIdentifier																																		
PublicCommentRecordBuffer (variable)																																		
...																																		

CommentIdentifier (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.

PublicCommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of public comment record. This SHOULD be one of the values listed in the preceding table, which are specified in the [EmrComment enumeration \(section 2.1.10\)](#), unless additional public comment record types have been implemented on the [print server](#).

PublicCommentRecordBuffer (variable): An optional array of bytes that contains the remainder of the public comment record. The size of this field MUST be a multiple of 4 bytes.

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
PublicCommentRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

PublicCommentRecordParm (variable): An optional array of bytes that contains the parameters for the public comment record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3.3](#) for additional comment record types.

2.3.3.4.1 EMR_COMMENT_BEGINGROUP Record

The EMR_COMMENT_BEGINGROUP record specifies the beginning of a group of drawing records.

Note Fields that are not described in this section are specified in section [2.3.3](#) or [2.3.3.4](#).

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
Type																																		
Size																																		
DataSize																																		
CommentIdentifier																																		
PublicCommentIdentifier																																		
Rectangle																																		
...																																		

...
...
nDescription
Description (variable)
...

PublicCommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of public comment record. This MUST be **EMR_COMMENT_BEGINGROUP** from the [EmrComment enumeration \(section 2.1.10\)](#), which is 0x00000002.

Rectangle (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that 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 additional public comment record types.

2.3.3.4.2 EMR_COMMENT_ENDGROUP Record

The EMR_COMMENT_ENDGROUP record specifies the end of a group of drawing records.

Note Fields that are not described in this section are specified in section [2.3.3](#) or [2.3.3.4](#).

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	3	0	1
Type																																
Size																																
DataSize																																
CommentIdentifier																																
PublicCommentIdentifier																																

PublicCommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of public comment record. This MUST be **EMR_COMMENT_ENDGROUP** from the [EmrComment enumeration \(section 2.1.10\)](#), 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 additional public comment record types.

2.3.3.4.3 EMR_COMMENT_MULTIFORMATS Record

The EMR_COMMENT_MULTIFORMATS record specifies an image in multiple graphics formats.

Note Fields that are not described in this section are specified in section [2.3.3](#) or [2.3.3.4](#).

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
Type																																		
Size																																		
DataSize																																		
CommentIdentifier																																		
PublicCommentIdentifier																																		
OutputRect																																		
...																																		
...																																		
...																																		
CountFormats																																		
aFormats (variable)																																		
...																																		
FormatData (variable)																																		
...																																		

PublicCommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of public comment record. This MUST be **EMR_COMMENT_MULTIFORMATS** from the [EmrComment enumeration \(section 2.1.10\)](#), which is 0x40000004.

OutputRect (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that specifies the output rectangle, in logical coordinates.

CountFormats (4 bytes): A 32-bit unsigned integer that specifies the number of graphics formats contained in this record.

aFormats (variable): A **CountFormats** length array of graphics formats, specified by [EmrFormat objects \(section 2.2.4\)](#), in order of preference.

FormatData (variable): A variable-length array of bytes of image data for all graphics formats contained in this record.

The size of the data for each image is provided by the **ContentSize** field in the corresponding EmrFormat object. Thus, the total size of this field is the sum of **ContentSize** values in all EmrFormat objects.

The graphics format of the data for each image is specified by the **Signature** field in the corresponding EmrFormat object.

For example, an application can use this record type to specify an image in EPS format using [EpsData objects \(section 2.2.6\)](#). Subsequently, the PostScript version of the image MAY [<77>](#) be rendered if that graphics format is supported by the printer driver on the playback system.

See section [2.3.3.4](#) for additional public comment record types.

2.3.3.4.4 EMR_COMMENT_WINDOWS_METAFILE Record

The EMR_COMMENT_WINDOWS_METAFILE record specifies an image in an embedded WMF metafile.

Note Fields that are not described in this section are specified in section [2.3.3](#) or [2.3.3.4](#).

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													
Type																																															
Size																																															
ContentSize																																															
CommentIdentifier																																															
PublicCommentIdentifier																																															
Version																Reserved																															
Checksum																																															
Flags																																															
WinMetafileSize																																															
WinMetafile (variable)																																															

...

PublicCommentIdentifier (4 bytes): A 32-bit unsigned integer that identifies the type of public comment record. This MUST be **EMR_COMMENT_WINDOWS_METAFILE** from the [EmrComment enumeration \(section 2.1.10\)](#), which is 0x80000001.

Version (2 bytes): A 16-bit unsigned integer that specifies the WMF metafile version in terms of support for device-independent bitmaps (DIBs), from the WMF [MetafileVersion](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.19).

Reserved (2 bytes): A 16-bit value that MUST be 0x0000 and MUST be ignored.

Checksum (4 bytes): A 32-bit unsigned integer that specifies the checksum for this record.

Flags (4 bytes): A 32-bit value that MUST be 0x00000000 and MUST be ignored.

WinMetafileSize (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the WMF metafile in the **WinMetafile** field.

WinMetafile (variable): A buffer that contains the WMF metafile.

See section [2.3.3.4](#) for additional public comment record types.

2.3.4 Control Record Types

The 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	2.3.4.1	Indicates the end of the metafile and specifies a palette.
EMR_HEADER	2.3.4.2	Indicates the start of the metafile and specifies properties of the device on which the metafile was created.

The generic structure of control records is specified as follows.

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
Type																																		
Size																																		
RecordBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The control record types are listed in the following table. See the preceding table 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 this record in the metafile. This value MUST be a multiple of 4 bytes.

RecordBuffer (variable): An array of bytes that contains the remainder of the control record. The size of this field MUST be a multiple of 4 bytes.

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
ControlRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

ControlRecordParm (variable): An array of bytes that contains the parameters for the control record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.4.1 EMR_EOF Record

The EMR_EOF record indicates the end of the metafile and specifies a palette.

Note Fields that are not described in this section are specified in section [2.3.4](#).

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
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.

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, which is not required to be contiguous with the fixed portion of the EMR_EOF record. Accordingly, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored. The size of this field MUST be a multiple of 4 bytes.

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
UndefinedSpace1 (variable)																																		
...																																		
PaletteEntries (variable)																																		
...																																		
UndefinedSpace2 (variable)																																		
...																																		

PaletteEntries (variable): An array of [LogPaletteEntry objects \(section 2.2.18\)](#) that specifies the palette data.

SizeLast (4 bytes): A 32-bit unsigned integer that MUST be the same as **Size** and MUST be the last field of the record and hence the metafile. LogPaletteEntry objects, if they exist, MUST precede this field.

See section [2.3.4](#) for additional control record types.

2.3.4.2 EMR_HEADER Record Types

The EMR_HEADER record types define the starting points of EMF metafiles and specify properties of the device on which the image in the metafile was created. The information in the header record makes it possible for EMF metafiles to be independent of any specific output device.

The following are EMR_HEADER record types.

Name	Section	Description
EmfMetafileHeader	2.3.4.2.1	The header record present in the original version of EMF metafiles. <78>
EmfMetafileHeaderExtension1	2.3.4.2.2	The header record used in the first extension to EMF metafiles. <79>
EmfMetafileHeaderExtension2	2.3.4.2.3	The header record used in the second extension to EMF metafiles. <80>

The generic structure of **EMR_HEADER** records is specified as follows.

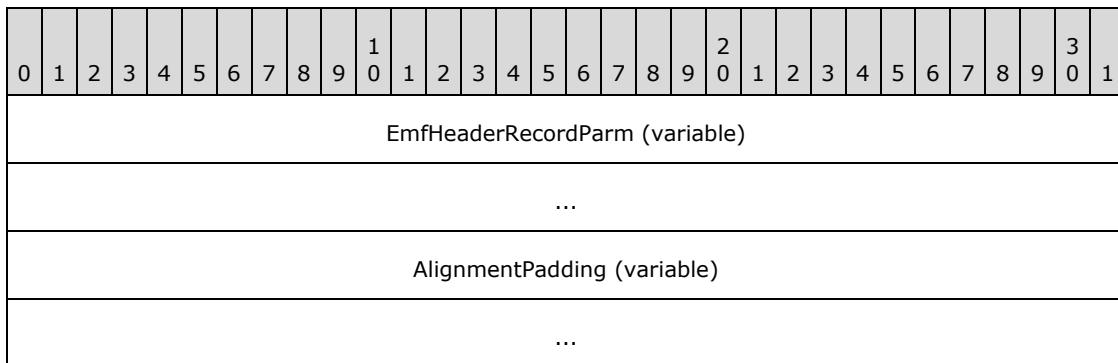
Note Fields that are not described in this section are specified in section [2.3.4](#).

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
Type																																		
Size																																		
EmfHeader																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
(EmfHeader cont'd for 12 rows)																																		
EmfHeaderRecordBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_HEADER**. This MUST be 0x00000001.

EmfHeader (80 bytes): A [Header](#) object (section [2.2.9](#)), which contains information about the content and structure of the metafile.

EmfHeaderRecordBuffer (variable): An optional array of bytes that contains the remainder of the EMF header record. The size of this field MUST be a multiple of 4 bytes.



EmfHeaderRecordParm (variable): An optional array of bytes that contains additional parameters for the EMF header record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

The value of the **Size** field can be used to distinguish between the different EMR_HEADER record types listed earlier in this section. There are three possible headers:

- The base header, which is the EmfMetafileHeader record. The fixed-size part of this header is 88 bytes, and it contains a Header object.
- The first extension header, which is the EmfMetafileHeaderExtension1 record. The fixed-size part of this header is 100 bytes, and it contains a Header object and a [HeaderExtension1](#) object (section [2.2.10](#)).
- The second extension header, which is the EmfMetafileHeaderExtension2 record. The fixed-size part of this header is 108 bytes, and it contains a Header object, a HeaderExtension1 object, and a [HeaderExtension2](#) object (section [2.2.11](#)).

Note that there are one or two optional, variable-length fields that are possible in a given header: a description string and a pixel format field. In all three types of headers, the fixed-size part comes first, followed by the variable-length fields.

The algorithm shown in the following figure computes a non-negative integer variable called **HeaderSize** from the offsets and lengths of the variable-length data. The type of header is determined from that value.

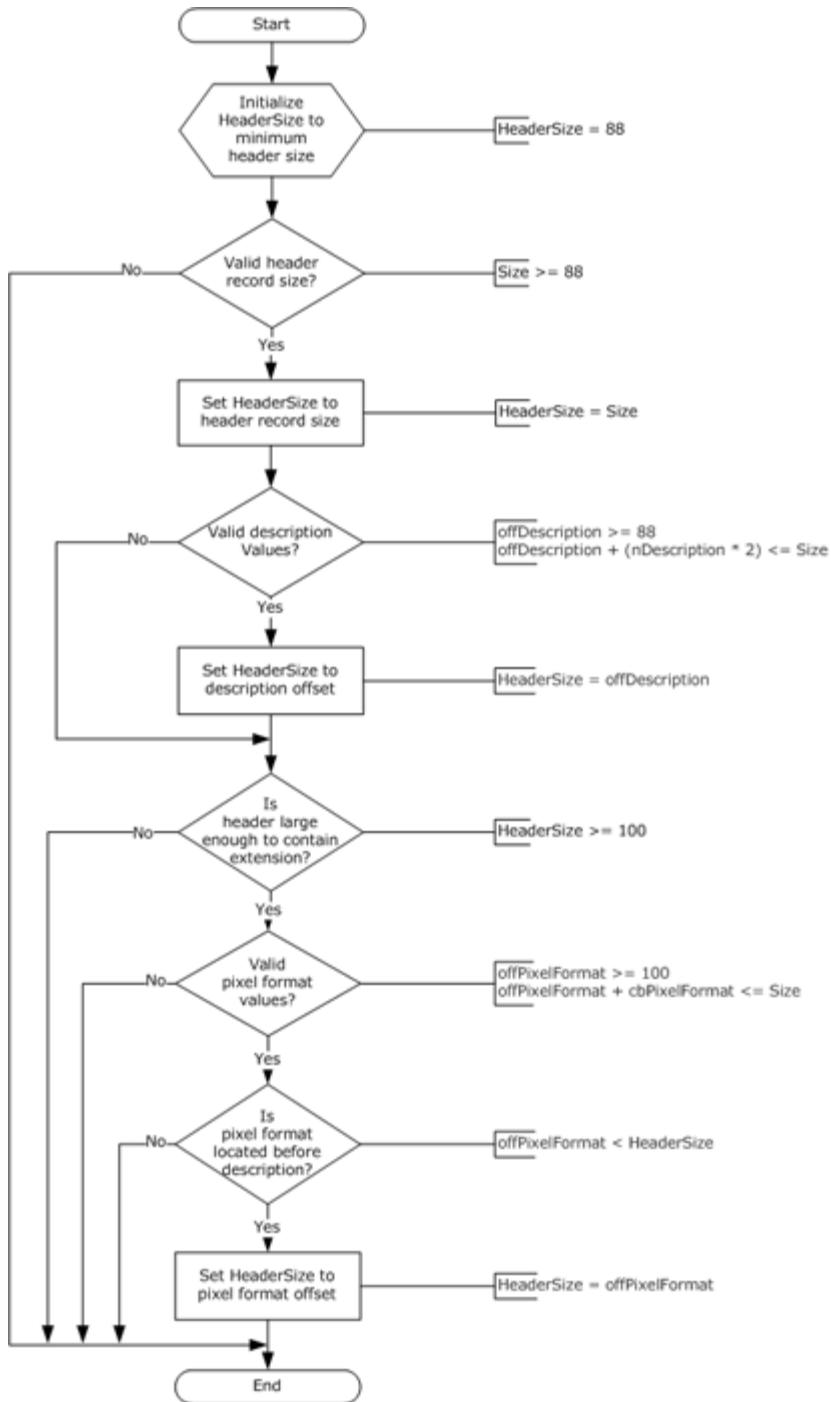


Figure 2: Header Type Determination Algorithm

After applying the algorithm, consider the value of **HeaderSize**:

- If **HeaderSize** ≥ 108 , the record type is EmfMetafileHeaderExtension2.

- If **HeaderSize** >= 100, the record type is EmfMetafileHeaderExtension1.
- Otherwise, the record type is EmfMetafileHeader.

See section [2.3.4](#) for additional control record types.

2.3.4.2.1 EmfMetafileHeader Record

The EmfMetafileHeader record is the header record used in the original version of EMF metafiles.

Note Fields that are not described in this section are specified in section [2.3.4](#) or [2.3.4.2](#).

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
Type																																		
Size																																		
EmfHeader																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
(EmfHeader cont'd for 12 rows)																																		
EmfDescriptionBuffer (variable)																																		
...																																		

EmfDescriptionBuffer (variable): An optional array of bytes that contains the EMF description string, which is not required to be contiguous with the fixed portion of the EmfMetafileHeader record. Accordingly, the field in this buffer that is labeled "UndefinedSpace" is optional and MUST be ignored.

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
UndefinedSpace (variable)																																		
...																																		
EmfDescription (variable)																																		
...																																		

EmfDescription (variable): An optional, null-terminated Unicode UTF16-LE string of arbitrary length and content. Its location in the record and number of characters are specified by the **offDescription** and **nDescription** fields, respectively, in **EmfHeader**. If the value of either field is zero, no description string is present.

The value of the **Size** field can be used to distinguish between the different EMR_HEADER record types. See EMR_HEADER record types (section 2.3.4.2) for details.

See section [2.3.4.2](#) for additional header record types.

2.3.4.2.2 EmfMetafileHeaderExtension1 Record

The EmfMetafileHeaderExtension1 record is the header record used in the first extension to EMF metafiles.[81](#) Following the **EmfHeaderExtension1** field, the remaining fields are optional and can be present in any order.

Note Fields that are not described in this section are specified in section [2.3.4](#) or [2.3.4.2](#).

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
Type																																		
Size																																		
EmfHeader																																		
...																																		
...																																		
...																																		
...																																		

...
(EmfHeader cont'd for 12 rows)
EmfHeaderExtension1
...
...
EmfDescriptionBuffer (variable)
...
EmfPixelFormatBuffer (variable)
...

EmfHeaderExtension1 (12 bytes): A [HeaderExtension1](#) object, which specifies additional information about the image in the metafile.

EmfDescriptionBuffer (variable): An optional array of bytes that contains the EMF description string, which is not required to be contiguous with the fixed portion of the EmfMetafileHeaderExtension1 record. Accordingly, the field in this buffer that is labeled "UndefinedSpace" is optional and MUST be ignored.

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	1
UndefinedSpace1 (variable)																																	
...																																	
EmfDescription (variable)																																	
...																																	

EmfDescription (variable): An optional, null-terminated Unicode UTF16-LE string of arbitrary length and content. Its location in the record and number of characters are specified by the **offDescription** and **nDescription** fields, respectively, in **EmfHeader**. If the value of either field is zero, no description string is present.

EmfPixelFormatBuffer (variable): An optional array of bytes that contains the EMF pixel format descriptor, which is not required to be contiguous with the fixed portion of the EmfMetafileHeaderExtension1 record or with the EMF description string. Accordingly, the field in this buffer that is labeled "UndefinedSpace" is optional and MUST be ignored.

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
UndefinedSpace2 (variable)																																		
...																																		
EmfPixelFormat (optional)																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
(EmfPixelFormat (optional) cont'd for 2 rows)																																		

EmfPixelFormat (40 bytes): An optional [PixelFormatDescriptor](#) object (section [2.2.22](#)) that specifies the last pixel format that was defined when the metafile was recorded. Its size and location in the record are specified by the **cbPixelFormat** and **offPixelFormat** fields, respectively, in **EmfHeaderExtension1**. If the value of either field is zero, no pixel format descriptor is present.

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.

The value of the **Size** field can be used to distinguish between the different EMR_HEADER record types. See EMR_HEADER record types (section 2.3.4.2) for details.

See section [2.3.4.2](#) for additional header record types.

2.3.4.2.3 EmfMetafileHeaderExtension2 Record

The EmfMetafileHeaderExtension2 record is the header record used in the second extension to EMF metafiles.[<82>](#) Following the **EmfHeaderExtension2** field, the remaining fields are optional and can be present in any order.

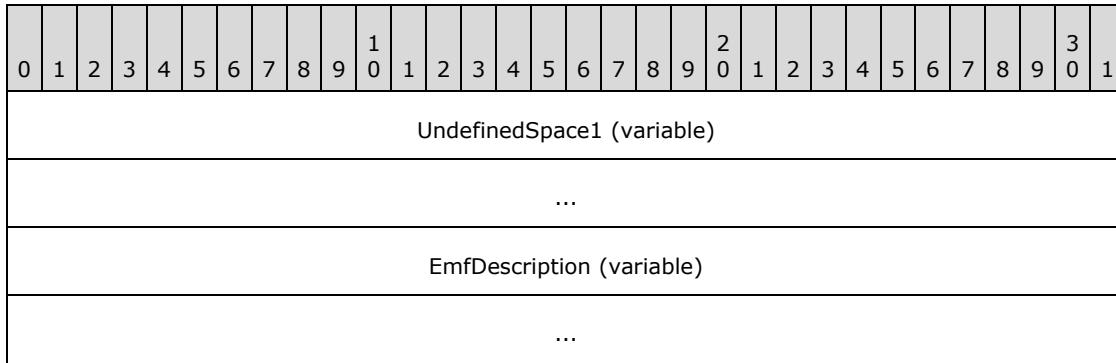
Note Fields that are not described in this section are specified in section [2.3.4](#) or [2.3.4.2](#).

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
Type																																		
Size																																		
EmfHeader																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		
(EmfHeader cont'd for 12 rows)																																		
EmfHeaderExtension1																																		
...																																		
...																																		
EmfHeaderExtension2																																		
...																																		
EmfDescriptionBuffer (variable)																																		
...																																		
EmfPixelFormatBuffer (variable)																																		
...																																		

EmfHeaderExtension1 (12 bytes): A [HeaderExtension1](#) object, which specifies additional information about the image in the metafile.

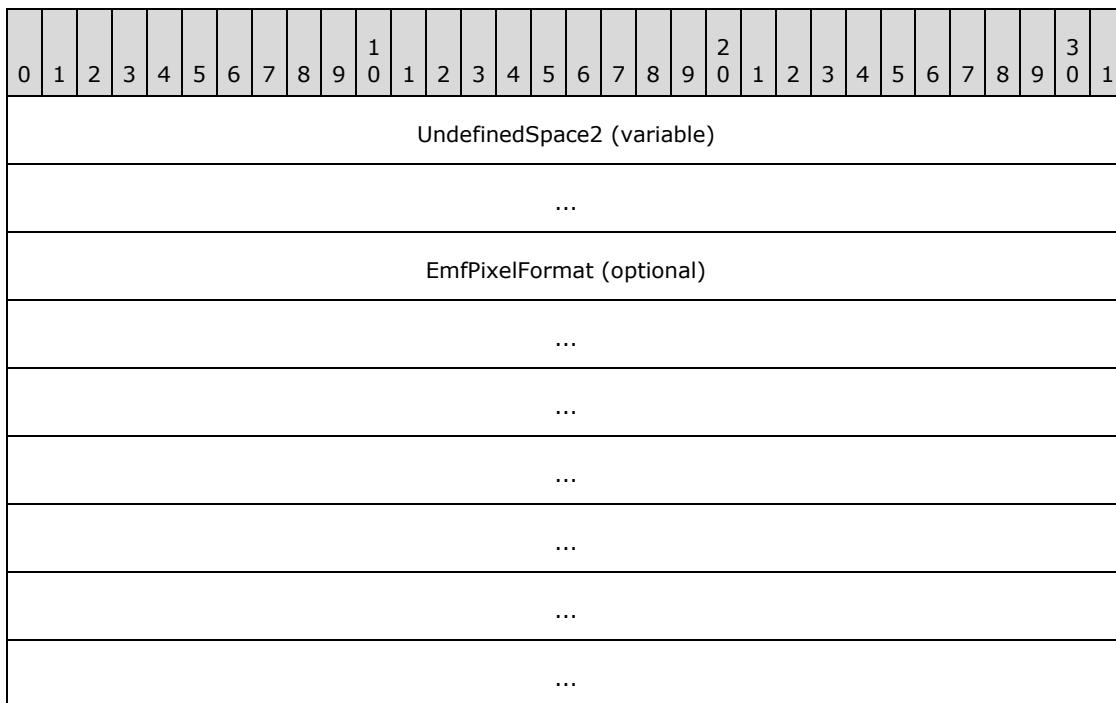
EmfHeaderExtension2 (8 bytes): A [HeaderExtension2](#) object, which specifies additional information about the image in the metafile.

EmfDescriptionBuffer (variable): An optional array of bytes that contains the EMF description string, which is not required to be contiguous with the fixed portion of the EmfMetafileHeaderExtension2 record. Accordingly, the field in this buffer that is labeled "UndefinedSpace" is optional and MUST be ignored.



EmfDescription (variable): An optional, null-terminated Unicode UTF16-LE string of arbitrary length and content. Its location in the record and number of characters are specified by the **offDescription** and **nDescription** fields, respectively, in **EmfHeader**. If the value of either field is zero, no description string is present.

EmfPixelFormatBuffer (variable): An optional array of bytes that contains the EMF pixel format descriptor, which is not required to be contiguous with the fixed portion of the EmfMetafileHeaderExtension2 record or with the EMF description string. Accordingly, the field in this buffer that is labeled "UndefinedSpace" is optional and MUST be ignored.



...
(EmfPixelFormat (optional) cont'd for 2 rows)

EmfPixelFormat (40 bytes): An optional [PixelFormatDescriptor](#) object (section [2.2.22](#)) that specifies the last pixel format that was defined when the metafile was recorded. Its size and location in the record are specified by the **cbPixelFormat** and **offPixelFormat** fields, respectively, in **EmfHeaderExtension1**. If the value of either field is zero, no pixel format descriptor is present.

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.

The value of the **Size** field can be used to distinguish between the different EMR_HEADER record types. See EMR_HEADER record types (section 2.3.4.2) for details.

See section [2.3.4.2](#) for additional header record types.

2.3.5 Drawing Record Types

The drawing record types perform graphics drawing.

The following are EMF drawing record types.

Name	Section	Description
EMR_ANGLEARC	2.3.5.1	Draws a line segment of an arc.
EMR_ARC	2.3.5.2	Draws an elliptical arc.
EMR_ARCTO	2.3.5.3	Draws an elliptical arc, resetting the current drawing position to the end point of the arc.
EMR_CHORD	2.3.5.4	Draws a chord, which is a region bounded by the intersection of an ellipse and a line segment, called a secant.
EMR_ELLIPSE	2.3.5.5	Draws an ellipse.
EMR_EXTFLOODFILL	2.3.5.1	Draws a line segment of an arc.
EMR_EXTTEXTOUTA	2.3.5.7	Draws an ASCII text string using the current font and text colors.
EMR_EXTTEXTOUTW	2.3.5.8	Draws a Unicode text string using the current font and text colors.
EMR_FILLPATH	2.3.5.9	Closes any open figures in the current path and fills the path's interior with the current brush and polygon-filling mode.
EMR_FILLRGN	2.3.5.10	Fills the specified region with the specified brush.
EMR_FRAMERGN	2.3.5.11	Draws a border around the specified region with the specified brush.

Name	Section	Description
EMR_GRADIENTFILL	2.3.5.12	Fills the specified rectangle and triangle structures.
EMR_LINETO	2.3.5.13	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	2.3.5.14	Paints the specified region with the current brush.
EMR_PIE	2.3.5.15	Draws a pie-shaped wedge bounded by the intersection of an ellipse and two radials.
EMR_POLYBEZIER	2.3.5.16	Draws one or more Bezier curves. The cubic Bezier curves are defined with the endpoints and control points specified in this record.
EMR_POLYBEZIER16	2.3.5.17	Draws one or more Bezier curves with the current pen.
EMR_POLYBEZIERTO	2.3.5.18	Draws one or more Bezier curves based on the current position.
EMR_POLYBEZIERTO16	2.3.5.19	Draws one or more Bezier curves based on the current position.
EMR_POLYDRAW	2.3.5.20	Draws a set of line segments and Bezier curves.
EMR_POLYDRAW16	2.3.5.21	Draws a set of line segments and Bezier curves.
EMR_POLYGON	2.3.5.22	Draws a polygon consisting of two or more vertexes connected by straight lines.
EMR_POLYGON16	2.3.5.23	Draws a polygon consisting of two or more vertexes connected by straight lines.
EMR_POLYLINE	2.3.5.24	Draws a series of line segments by connecting the points in the specified array.
EMR_POLYLINE16	2.3.5.25	Draws a series of line segments by connecting the points in the specified array.
EMR_POLYLINETO	2.3.5.26	Draws one or more straight lines based upon the current position.
EMR_POLYLINETO16	2.3.5.27	Draws one or more straight lines based upon the current position.
EMR_POLYPOLYGON	2.3.5.28	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	2.3.5.29	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	2.3.5.30	Draws multiple series of connected line segments.
EMR_POLYPOLYLINE16	2.3.5.31	Draws multiple series of connected line segments.
EMR_POLYTEXTOUTA	2.3.5.32	Draws one or more ASCII text strings using the current font and text colors.

Name	Section	Description
EMR_POLYTEXTOUTW	2.3.5.33	Draws one or more Unicode text strings using the current font and text colors.
EMR_RECTANGLE	2.3.5.34	Draws a rectangle. The rectangle is outlined with the current pen and filled with the current brush.
EMR_ROUNDRECT	2.3.5.35	Draws a rectangle with rounded corners.
EMR_SETPIXELV	2.3.5.36	Defines the color of the pixel at the specified logical coordinates.
EMR_SMALLTEXTOUT	2.3.5.37	Outputs a string.
EMR_STROKEANDFILLPATH	2.3.5.38	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	2.3.5.39	Draws the specified path with the current pen.

The generic structure of drawing records is specified as follows.

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
Type																																		
Size																																		
DrawingRecordBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The drawing record types are listed in the following table. See the preceding table 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

Name	Value
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
EMR_STROKEPATH	0x00000040
EMR_FILLRGN	0x00000047
EMR_FRAMERGN	0x00000048
EMR_PAINTRGN	0x0000004A
EMR_EXTPTEXTOUTA	0x00000053
EMR_EXTPTEXTOUTW	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 this record in the metafile. This value MUST be a multiple of 4 bytes.

DrawingRecordBuffer (variable): An array of bytes that contains the remainder of the drawing record. The size of this field MUST be a multiple of 4 bytes.

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
DrawingRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

DrawingRecordParm (variable): An array of bytes that contains the parameters for the drawing record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.5.1 EMR_ANGLEARC Record

The 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	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
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 WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, which 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 EMR_ARC record specifies an elliptical arc.

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
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 2.2.2.19, which specifies the inclusive-inclusive bounding rectangle.

Start (8 bytes): A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, 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 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	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
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-bit WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which specifies the bounding rectangle.

Start (8 bytes): A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, which specifies the coordinates of the first radial ending point, in logical units.

End (8 bytes): A 64-bit WMF **PointL** object that specifies the coordinates of the 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 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which specifies the inclusive-inclusive bounding rectangle.

Start (8 bytes): A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, 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 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	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
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 2.2.2.19, 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 EMR_EXTFLOODFILL record fills an area of the display surface with the current brush.

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
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 WMF **PointL** object ([\[MS-WMF\]](#) section 2.2.2.15), which specifies the coordinates, in logical units, where filling begins.

Color (4 bytes): A WMF **ColorRef** object ([\[MS-WMF\]](#) section 2.2.2.8), which is used with the **FloodFillMode** to determine the area to fill.

FloodFillMode (4 bytes): A 32-bit unsigned integer that specifies how to use the **Color** value to determine the area for the flood fill operation. The value MUST be in the [FloodFill](#) enumeration (section [2.1.13](#)).

See section [2.3.5](#) for the specification of other Drawing record types.

2.3.5.7 EMR_EXTEXTOUTA Record

The EMR_EXTEXTOUTA record draws an ASCII text string using the current font and text colors.

Note Fields that are not described in this section are specified in section [2.3.5](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
iGraphicsMode																																		
exScale																																		

eyScale
aEmrText (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies the record type as **EMR_EXTTEXTOUTA** from the [RecordType enumeration \(section 2.1.1\)](#). This MUST be 0x00000053.

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19). It is not used and MUST be ignored on receipt.

iGraphicsMode (4 bytes): A 32-bit unsigned integer that specifies the graphics mode from the [GraphicsMode enumeration \(section 2.1.16\)](#).

exScale (4 bytes): A 32-bit floating-point value that specifies the scale factor to apply along the X axis to convert from page space units to .01mm units. This SHOULD be used only if the graphics mode specified by **iGraphicsMode** is **GM_COMPATIBLE**.

eyScale (4 bytes): A 32-bit floating-point value that specifies the scale factor to apply along the Y axis to convert from page space units to .01mm units. This SHOULD be used only if the graphics mode specified by **iGraphicsMode** is **GM_COMPATIBLE**.

aEmrText (variable): An [EmrText object \(section 2.2.5\)](#) that specifies the output string in 8-bit ASCII characters, text attributes, and spacing values.

The font and text colors used for output are specified by properties in the current state of the playback device context. A rectangle for clipping and/or opaquing can be defined in the EmrText object that is specified in the **aEmrText** field.

EMR_EXTTEXTOUTA SHOULD be emulated with an **EMR_EXTTEXTOUTW** record (section [2.3.5.8](#)). This requires the ASCII text string in the EmrText object to be converted to Unicode UTF16-LE encoding.[<83>](#)

See section [2.3.5](#) for additional drawing record types.

2.3.5.8 EMR_EXTTEXTOUTW Record

The EMR_EXTTEXTOUTW record draws a Unicode text string using the current font and text colors.

Note Fields that are not described in this section are specified in section [2.3.5](#).

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
Type																																		
Size																																		
Bounds																																		

...
...
...
iGraphicsMode
exScale
eyScale
wEmrText (variable)
...

Type (4 bytes): A 32-bit unsigned integer that identifies the record type as **EMR_EXTTEXTOUTW** from the [RecordType enumeration \(section 2.1.1\)](#). This value MUST be 0x00000054.

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19). It is not used and MUST be ignored on receipt.

iGraphicsMode (4 bytes): A 32-bit unsigned integer that specifies the current graphics mode from the [GraphicsMode enumeration \(section 2.1.16\)](#).

exScale (4 bytes): A 32-bit floating-point value that specifies the scale factor to apply along the X axis to convert from page space units to .01mm units. This is used only if the graphics mode specified by **iGraphicsMode** is **GM_COMPATIBLE**.

eyScale (4 bytes): A 32-bit floating-point value that specifies the scale factor to apply along the Y axis to convert from page space units to .01mm units. This is used only if the graphics mode specified by **iGraphicsMode** is **GM_COMPATIBLE**.

wEmrText (variable): An [EmrText object \(section 2.2.5\)](#) that specifies the output string in 16-bit Unicode UTF16-LE characters, with text attributes and spacing values.

The font and text colors used for output are specified by properties in the current state of the playback device context. A rectangle for clipping and/or opaquning can be defined in the EmrText object that is specified in the **aEmrText** field.

See section [2.3.5](#) for additional drawing record types.

2.3.5.9 EMR_FILLPATH Record

The 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	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
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 2.2.2.19, 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 EMR_FILLRGN record fills the specified region by using the specified brush.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, 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 to fill the region.

RgnData (variable): A **RgnDataSize** length array of bytes that contains a [RegionData \(section 2.2.24\)](#) object.

See section [2.3.5](#) for the specification of other Drawing record types.

2.3.5.11 EMR_FRAMERGN Record

The EMR_FRAMERGN record draws a border around the specified region using the specified brush.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, 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 EMR_GRADIENTFILL record specifies filling rectangles or triangles with gradients of color. [<84>](#)

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) that specifies a bounding rectangle, in inclusive-inclusive 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 to fill.

ulMode (4 bytes): A 32-bit unsigned integer that specifies the gradient fill mode. The value MUST be in the [GradientFill enumeration \(section 2.1.15\)](#).

VertexData (variable): Objects that specify the vertexes of either rectangles or triangles and the colors that correspond to them.

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
VertexObjects (variable)																																		
...																																		
VertexIndexes (variable)																																		
...																																		
VertexPadding (variable)																																		
...																																		

VertexObjects (variable): An array of **nVer** [TriVertex objects \(section 2.2.26\)](#). Each object specifies the position and color of a vertex of either a rectangle or a triangle, depending on the value of the **ulMode** field.

VertexIndexes (variable): An array of **nTri** [GradientRectangle objects \(section 2.2.7\)](#) or [GradientTriangle objects \(section 2.2.8\)](#), depending on the value of the **ulMode** field. Each object specifies indexes into the array of TriVertex objects in the **VertexObjects** field.

VertexPadding (variable): An optional variable-length array of **nTri** times four bytes that MUST be present if the value of the **ulMode** field indicates GradientRectangle

objects (section 2.2.7). If the value of the **uIMode** field indicates GradientTriangle objects (section 2.2.8), no **VertexPadding** is present. This field MUST be ignored.

An EMR_GRADIENTFILL record that specifies that the three vertexes of a triangle SHOULD fill the figure with smooth gradients of colors.[<85>](#)

An EMR_GRADIENTFILL record that specifies that the upper-left and lower-right vertexes of a rectangle SHOULD fill the figure with smooth gradients of color. There are two gradient fill modes in the **GradientFill** enumeration that can be used when drawing a rectangle. In **GRADIENT_FILL_RECT_H** mode, the rectangle is filled from left to right. In **GRADIENT_FILL_RECT_V** mode, the rectangle is filled from top to bottom.

Note An EMR_GRADIENTFILL record MUST ignore the **Alpha** fields in the TriVertex objects. An [EMR_ALPHABLEND record \(section 2.3.1.1\)](#) that immediately follows the EMR_GRADIENTFILL record can be used to apply an alpha transparency gradient to the filled area.

See section [2.3.5](#) for the specification of other Drawing record types.

2.3.5.13 EMR_LINETO Record

The 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	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
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 WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, which 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 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, 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 \(section 2.2.24\)](#) 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 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	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	1
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-bit WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which specifies the inclusive-inclusive bounding rectangle.

Start (8 bytes): A 64-bit WMF **PointL** objects, specified in [\[MS-WMF\]](#) section 2.2.2.15, which specifies the coordinates, in logical units, of the endpoint of the first radial.

End (8 bytes): A 64-bit **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 EMR_POLYBEZIER record specifies one or more Bezier curves.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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 2.2.2.15) that specifies 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 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYBEZIERTO record specifies one or more Bezier curves based upon the current position.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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 2.2.2.15) that specifies 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 EMR_POLYBEZIERTO16 record specifies one or more Bezier curves based on the current position.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which specifies 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 EMR_POLYDRAW record specifies a set of line segments and Bezier curves.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.15, which specifies the array of points, in logical units.

abTypes (variable): A **Count** length array of byte 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 EMR_POLYDRAW16 record specifies a set of line segments and Bezier curves.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYGON record specifies a polygon consisting of two or more vertexes connected by straight lines.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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 2.2.2.15) that specifies the vertexes 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 EMR_POLYGON16 record specifies 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.

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	1
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 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	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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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 2.2.2.15) that specifies 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 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYLINETO record specifies one or more straight lines based upon the current position.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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

Line width	Device supports wideline	Maximum points allowed
> 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 2.2.2.15) that specifies 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_POLYLINETO16 Record

The EMR_POLYLINETO16 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	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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
Count																																		
aPoints (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_POLYLINETO16**. 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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYPOLYGON record specifies a series of closed polygons.

Note Fields that are not described in this section are specified in section [2.3.5](#).

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
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.

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) 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 an EMR_POLYPOLYGON operation SHOULD be performed for each group of points.

PolygonPointCount (variable): An array of 32-bit unsigned integers that specifies the point count for each polygon.

aPoints (variable): An array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.2.15) that specifies the points for all polygons in logical units. The number of points is specified by the **Count** field value.

Each polygon SHOULD be outlined using the current pen, and filled using the current brush and polygon fill mode that are defined in the playback device context. The polygons defined by this record can overlap.

See section [2.3.5](#) for additional drawing record types.

2.3.5.29 EMR_POLYPOLYGON16 Record

The 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYPOLYLINE record specifies multiple series of connected line segments.

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
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 WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19) that specifies the bounding rectangle in device units.

NumberOfPolylines (4 bytes): A 32-bit unsigned integer that specifies the number of polylines, which is the number of elements in the **aPolylinePointCount** array.

Count (4 bytes): A 32-bit unsigned integer that specifies the total number of points in all polylines, which is the number of elements 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.

aPolylinePointCount (variable): A **NumberOfPolylines**-length array of 32-bit unsigned integers that specify the point counts for all polylines. Each value MUST be greater than or equal to 0x00000002.

Each point count refers to a number of consecutive elements in the **aPoints** array.

aPoints (variable): A **Count**-length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.2.15) that 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 be filled. The current position SHOULD neither be 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 EMR_POLYPOLYLINE16 record specifies multiple series of connected line segments.

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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which 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 2.2.2.16, which 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 EMR_POLYTEXTOUTA record draws one or more ASCII text strings using the current font and text colors.

Note Fields that are not described in this section are specified in section [2.3.5](#).

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
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.

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19), which specifies the bounding rectangle in device units.

iGraphicsMode (4 bytes): A 32-bit unsigned integer that specifies the current graphics mode, from the **GraphicsMode** enumeration (section [2.1.16](#)).

exScale (4 bytes): A 32-bit floating-point value that specifies the X scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

eyScale (4 bytes): A 32-bit floating-point value 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 [EmrText](#) objects (section [2.2.5](#)) that specify the output strings in 8-bit ASCII characters, with text attributes, and spacing values. The number of EmrText objects is specified by **cStrings**.

The font and text colors used for output are specified by properties in the current state of the playback device context.

EMR_POLYTEXTOUTA SHOULD be emulated with a series of [EMR_EXTTEXTOUTW](#) records (section [2.3.5.7](#)), one per string. This requires the ASCII text string in each EmrText object to be converted to Unicode UTF16-LE encoding. [<86>](#)

See section [2.3.5](#) for additional drawing record types.

2.3.5.33 EMR_POLYTEXTOUTW Record

The EMR_POLYTEXTOUTW record draws one or more Unicode text strings using the current font and text colors.

Note Fields that are not described in this section are specified in section [2.3.5](#).

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
...																																		
iGraphicsMode																																		
exScale																																		
eyScale																																		
cStrings																																		
wEmrText (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_POLYTEXTOUTW**. This MUST be 0x00000061.

Bounds (16 bytes): A WMF **RectL** object ([\[MS-WMF\]](#) section 2.2.2.19), 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 floating-point value that specifies the X scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

eyScale (4 bytes): A 32-bit floating-point value 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.

wEmrText (variable): An array of [EmrText objects \(section 2.2.5\)](#) that specify the output strings in 16-bit Unicode UTF16-LE characters, with text attributes and spacing values. The number of EmrText objects is specified by **cStrings**.

The font and text colors used for output are specified by properties in the current state of the playback device context.

EMR_POLYTEXTOUTW SHOULD be emulated with a series of [EMR_EXTTEXTOUTW](#) records (section [2.3.5.7](#)), one per string.[<87>](#)

See section [2.3.5](#) for additional drawing record types.

2.3.5.34 EMR_RECTANGLE Record

The 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, 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 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	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
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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, which specifies the inclusive-inclusive bounding rectangle, in logical coordinates.

Corner (8 bytes): A 64-bit WMF **SizeL** object, specified in [\[MS-WMF\]](#) section 2.2.2.22, 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 EMR_SETPIXELV record defines the color of the pixel at the specified logical coordinates.

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
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 WMF [PointL](#) object ([\[MS-WMF\]](#) section 2.2.2.15) that specifies the logical coordinates for the pixel.

Color (4 bytes): A 32-bit WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8) 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 EMR_SMALLTEXTOUT record outputs a string.

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
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 ([\[MS-WMF\]](#) section 2.2.2.19) that specifies the bounding rectangle in device units.

TextString (variable): A variable-length string that contains the text string to draw, in either 8-bit 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 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	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
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 WMF [RectL](#) object ([\[MS-WMF\]](#) section 2.2.2.19) 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 EMR_STROKEPATH record renders the specified path by using the current pen.

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
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 WMF [RectL](#) object ([MS-WMF](#) section 2.2.2.19) that specifies the bounding rectangle in device units.

See section [2.3.5](#) for the specification of other Drawing record types.

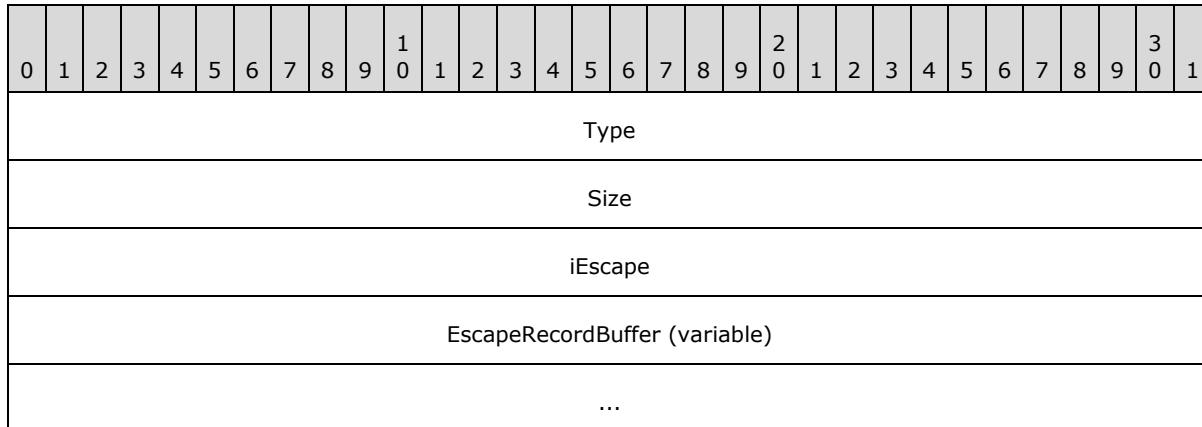
2.3.6 Escape Record Types

The escape record types execute printer driver functions.

The following are EMF escape record types.

Name	Section	Description
EMR_DRAWESCAPE	2.3.6.1	Passes arbitrary information to the printer driver. The intent is that the information will result in drawing being done.
EMR_EXTESCAPE	2.3.6.2	Passes arbitrary information to the printer driver. The intent is that the information will not result in drawing being done.
EMR_NAMEDESCAPE	2.3.6.3	Passes arbitrary information to the given named printer driver.

The generic structure of escape records is specified as follows.



Type (4 bytes): A 32-bit unsigned integer that defines the type of the record. The escape record types are listed in the following table. See the preceding table 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 in bytes of this record in the metafile. This value MUST be a multiple of 4 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 WMF [MetafileEscapes](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.17).

EscapeRecordBuffer (variable): An array of bytes that contains the remainder of the escape record. The size of this field MUST be a multiple of 4 bytes.

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
EscapeRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

EscapeRecordParm (variable): An array of bytes that contains the parameters for the escape record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.6.1 EMR_DRAWESCAPE Record

The 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 in this section are specified in section [2.3.6](#).

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
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 additional escape record types.

2.3.6.2 EMR_EXTEscape Record

The 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 in this section are specified in section [2.3.6](#).

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
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 additional escape record types.

2.3.6.3 EMR_NAMEDESCAPE Record

The MR_NAMEDESCAPE record passes arbitrary information to a specified printer driver.

Note Fields that are not described in this section are specified in section [2.3.6](#).

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
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 specifies 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 additional escape record types.

2.3.7 Object Creation Record Types

The object creation record types create graphics objects.

The following are EMF object creation record types.

Name	Section	Description
EMR_CREATEBRUSHINDIRECT	2.3.7.1	Defines a logical brush with a LogBrushEx object (section 2.2.12).
EMR_CREATECOLORSPACE	2.3.7.2	Defines a logical color space with a WMF LogColorSpace object (MS-WMF section 2.2.2.11).

Name	Section	Description
EMR_CREATECOLORSPACEW	2.3.7.3	Defines a logical color space with a WMF LogColorSpaceW object ([MS-WMF] section 2.2.2.12).
EMR_CREATEDIBPATTERNBRUSHPT	2.3.7.4	Defines a pattern brush with a WMF DeviceIndependentBitmap object ([MS-WMF] section 2.2.2.9).
EMR_CREATEMONOBRUSH	2.3.7.5	Defines a monochrome pattern brush with a monochrome WMF DeviceIndependentBitmap object.
EMR_CREATEPALETTE	2.3.7.6	Defines a logical palette with a LogPalette object (section 2.2.17).
EMR_CREATEPEN	2.3.7.7	Defines a logical pen with a LogPen object (section 2.2.19).
EMR_EXTCREATEFONTINDIRECTW	2.3.7.8	Defines a logical font with either a LogFont or LogFontExDv object (sections 2.2.13 and 2.2.15 , respectively).
EMR_EXTCREATEPEN	2.3.7.9	Defines an extended logical pen with a LogPenEx object (section 2.2.20) and optional WMF DeviceIndependentBitmap object.

The generic structure of object creation records is specified as follows.

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
Type																																		
Size																																		
ObjectRecordBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The object creation record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_CREATEMONOBRUSH	0x0000005D
EMR_CREATEDIBPATTERNBRUSHPT	0x0000005E
EMR_EXTCREATEPEN	0x0000005F
EMR_CREATECOLORSPACEW	0x0000007A
EMR_CREATEPEN	0x00000026

Name	Value
EMR_CREATEBRUSHINDIRECT	0x00000027
EMR_CREATEPALETTE	0x00000031
EMR_EXTCREATEFONTINDIRECTW	0x00000052
EMR_CREATECOLORSPACE	0x00000063

Size (4 bytes): A 32-bit unsigned integer that specifies the size in bytes of this record in the metafile. This value MUST be a multiple of 4 bytes.

ObjectRecordBuffer (variable): An array of bytes that contains the remainder of the object creation record. The size of this field MUST be a multiple of 4 bytes.

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
ObjectRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

ObjectRecordParm (variable): An array of bytes that contains the parameters for the object creation record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.7.1 EMR_CREATEBRUSHINDIRECT Record

The EMR_CREATEBRUSHINDIRECT record defines a logical brush for graphics operations.

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
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, in bytes, of this record. This MUST be 0x00000018.

ihBrush (4 bytes): A 32-bit unsigned integer that specifies the index of the logical brush object in the [EMF Object Table](#) (section 3.1.1.1). This index MUST be saved so that this object can be reused or modified.

LogBrush (12 bytes): A [LogBrushEx](#) object (section 2.2.12) that specifies the style, color, and pattern of the logical brush. The **BrushStyle** field in this object MUST be **BS_SOLID**, **BS_HATCHED**, or **BS_NULL**.

The logical brush object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section 2.3.8.5), which specifies the logical brush to use in subsequent graphics operations.

See section 2.3.7 for additional object creation record types.

2.3.7.2 EMR_CREATECOLORSPACE Record

The EMR_CREATECOLORSPACE record creates a logical color space object from a color profile with a name consisting of ASCII characters.[<88>](#)

Note Fields that are not described in this section are specified in section 2.3.7.

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
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.

ihCS (4 bytes): A 32-bit unsigned integer that specifies the index of the logical color space object in the [EMF object table](#) (section 3.1.1.1). This index MUST be saved so that this object can be reused or modified.

lcs (variable): A WMF **LogColorSpace** object ([\[MS-WMF\]](#) section 2.2.2.11), which can specify the name of a color profile in ASCII characters.

The logical color space object defined by this record can be selected into the playback device context by an [EMR_SETCOLORSPACE](#) record (section 2.3.8.7), which defines the logical color space to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.3 EMR_CREATECOLORSPACEW Record

The EMR_CREATECOLORSPACEW record creates a logical color space object from a color profile with a name consisting of Unicode characters.[89](#)

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
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.

ihCS (4 bytes): A 32-bit unsigned integer that specifies the index of the logical color space object in the [EMF object table](#) (section 3.1.1.1). This index MUST be saved so that this object can be reused or modified.

lcs (variable): A WMF **LogColorSpaceW** object ([\[MS-WMF\]](#) section 2.2.2.12) that can specify the name of a color profile in Unicode UTF16-LE characters.

dwFlags (4 bytes): A 32-bit unsigned integer that provides information about the data in this record.

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
0																																C		

C (1 bit): If set, the **Data** field contains color profile data.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the **Data** field.

Data (variable): An optional array of bytes that specifies color profile data.

The logical color space object defined by this record can be selected into the playback device context by an [EMR_SETCOLORSPACE](#) record (section [2.3.8.7](#)), which defines the logical color space to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.4 EMR_CREATEDIBPATTERNBRUSHPT Record

The EMR_CREATEDIBPATTERNBRUSHPT record defines a pattern brush for graphics operations. The pattern is specified by a DIB.

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
Type																																		
Size																																		
ihBrush																																		
Usage																																		
offBmi																																		
cbBmi																																		
offBits																																		
cbBits																																		
BitmapBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as EMR_CREATEDIBPATTERNBRUSHPT. This MUST be 0x0000005E.

ihBrush (4 bytes): A 32-bit unsigned integer that specifies the index of the pattern brush object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

Usage (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the DIB header. This value MUST be in the [DIBColors](#) enumeration (section [2.1.9](#)).

offBmi (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB header.

cbBmi (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB header.

offBits (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB bits.

cbBits (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB bits.

BitmapBuffer (variable): A buffer containing a packed DIB in the form of a WMF [DeviceIndependentBitmap](#) object ([MS-WMF](#) section 2.2.2.9). It is not required to be contiguous with the fixed portion of the EMR_CREATEDIBPATTERNBRUSHPT record.

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	1
UndefinedSpace (variable)																																	
...																																	
BmiSrc (variable)																																	
...																																	
BitsSrc (variable)																																	
...																																	

UndefinedSpace (variable): An optional array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a WMF [DeviceIndependentBitmap](#) object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a WMF [DeviceIndependentBitmap](#) object.

The pattern brush object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section 2.3.8.5), which specifies the pattern brush to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.5 EMR_CREATEMONOBRUSH Record

The EMR_CREATEMONOBRUSH record defines a monochrome pattern brush for graphics operations. The pattern is specified by a monochrome DIB.

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
Type																																		
Size																																		
ihBrush																																		
Usage																																		
offBmi																																		
cbBmi																																		
offBits																																		
cbBits																																		
BitmapBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_CREATEMONOBRUSH**. This MUST be 0x0000005D.

ihBrush (4 bytes): A 32-bit unsigned integer that specifies the index of the monochrome pattern brush object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

Usage (4 bytes): A 32-bit unsigned integer that specifies how to interpret values in the color table in the DIB header. This value MUST be in the [DIBColors](#) enumeration (section [2.1.9](#)).

offBmi (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB header.

cbBmi (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB header.

offBits (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB bits.

cbBits (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB bits.

BitmapBuffer (variable): A buffer containing a packed DIB in the form of a monochrome WMF [DeviceIndependentBitmap](#) object ([\[MS-WMF\]](#) section 2.2.2.9). It is not required to be contiguous with the fixed portion of the EMR_CREATEMONOBRUSH record.

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
UndefinedSpace (variable)																																		

...
BmiSrc (variable)
...
BitsSrc (variable)
...

UndefinedSpace (variable): An optional array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a WMF DeviceIndependentBitmap object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a WMF DeviceIndependentBitmap object.

The monochrome pattern brush object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)), which specifies the pattern brush to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.6 EMR_CREATEPALETTE Record

The EMR_CREATEPALETTE record defines a logical palette for graphics operations.

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
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.

ihPal (4 bytes): A 32-bit unsigned integer that specifies the index of the logical palette object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

LogPalette (variable): A [LogPalette](#) object (section [2.2.17](#)). The **Version** field of this object MUST be set to 0x0300. If the **NumberOfEntries** value in this object is zero, processing of this record MUST fail.

The logical palette defined by this record can be selected into the playback device context by an [EMR_SELECTPALETTE](#) record (section [2.3.8.6](#)), which specifies the logical palette to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.7 EMR_CREATEPEN Record

The EMR_CREATEPEN record defines a logical pen for graphics operations.

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
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, in bytes, of this record. This MUST be 0x0000001C.

ihPen (4 bytes): A 32-bit unsigned integer that specifies the index of the logical pen object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

LogPen (16 bytes): A [LogPen](#) object (section [2.2.19](#)) that specifies the style, width, and color of the logical pen.

The logical pen object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)), which specifies the logical pen to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

2.3.7.8 EMR_EXTCREATEFONTINDIRECTW Record

The EMR_EXTCREATEFONTINDIRECTW record defines a logical font for graphics operations.

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
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.

ihFonts (4 bytes): A 32-bit unsigned integer that specifies the index of the logical font object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

elw (variable): A [LogFontExDv](#) object (section [2.2.15](#)), which specifies the logical font. A [LogFont](#) object [2.2.13](#) MAY be present instead. [<90>](#) The process for determining the type of object in this field is described below.

The logical font object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)), which specifies the logical font to use in subsequent graphics operations.

The type of logical font object in the **elw** field of this record 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 follows.

Size - 12

- If the size of the **elw** field is equal to or less than the size of a [LogFontPanose](#) object (section [2.2.16](#)), **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 LogFontPanose object, then **elw** MUST be treated as a variable-length LogFontExDv object.

The size of a LogFontPanose object is 0x0140 (320 decimal). It is determined by adding up the sizes of its fields, as follows:

- LogFont:** The size of a LogFont object 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 [2.3.7](#) for additional object creation record types.

2.3.7.9 EMR_EXTCREATEPEN Record

The EMR_EXTCREATEPEN record defines an extended logical pen for graphics operations. An optional DIB can be specified to use as the line style.

Note Fields that are not described in this section are specified in section [2.3.7](#).

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
Type																																		
Size																																		
ihPen																																		
offBmi																																		
cbBmi																																		
offBits																																		
cbBits																																		
elp (variable)																																		
...																																		
BitmapBuffer (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_EXTCREATEPEN**. This MUST be 0x0000005F.

ihPen (4 bytes): A 32-bit unsigned integer that specifies the index of the extended logical pen object in the [EMF Object Table](#) (section [3.1.1.1](#)). This index MUST be saved so that this object can be reused or modified.

offBmi (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB header, if the record contains a DIB.

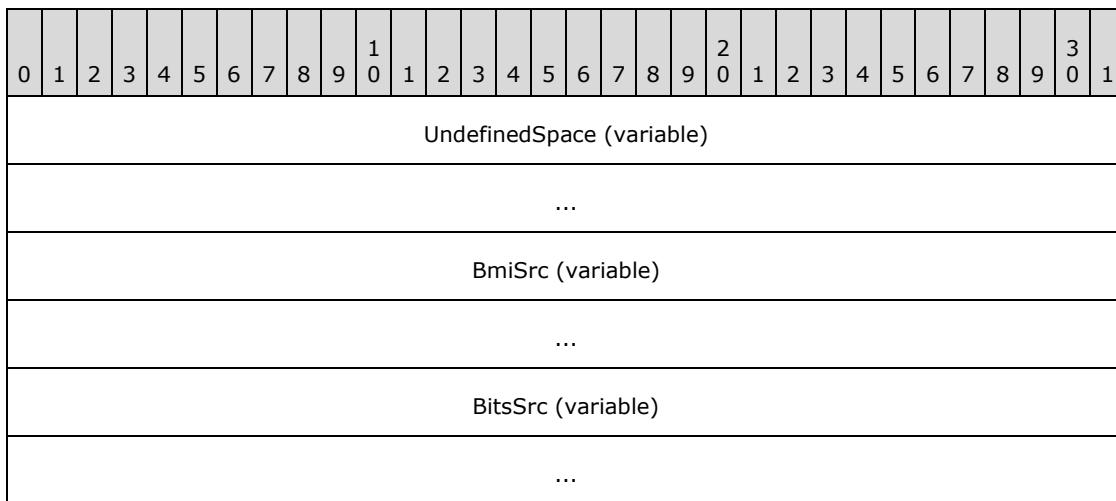
cbBmi (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB header, if the record contains a DIB.

offBits (4 bytes): A 32-bit unsigned integer that specifies the offset from the start of this record to the DIB bits, if the record contains a DIB.

cbBits (4 bytes): A 32-bit unsigned integer that specifies the size of the DIB bits, if the record contains a DIB.

elp (variable): A [LogPenEx](#) object (section [2.2.20](#)) that specifies an extended logical pen with attributes including an optional line style array.

BitmapBuffer (variable): An optional buffer containing a packed DIB in the form of a WMF [DeviceIndependentBitmap](#) object ([\[MS-WMF\]](#) section 2.2.2.9). It is not required to be contiguous with the fixed portion of the EMR_EXTCREATEOPEN record.



UndefinedSpace (variable): An optional array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a WMF [DeviceIndependentBitmap](#) object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a WMF [DeviceIndependentBitmap](#) object.

The extended logical pen object defined by this record can be selected into the playback device context by an [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)), which specifies the logical pen to use in subsequent graphics operations.

See section [2.3.7](#) for additional object creation record types.

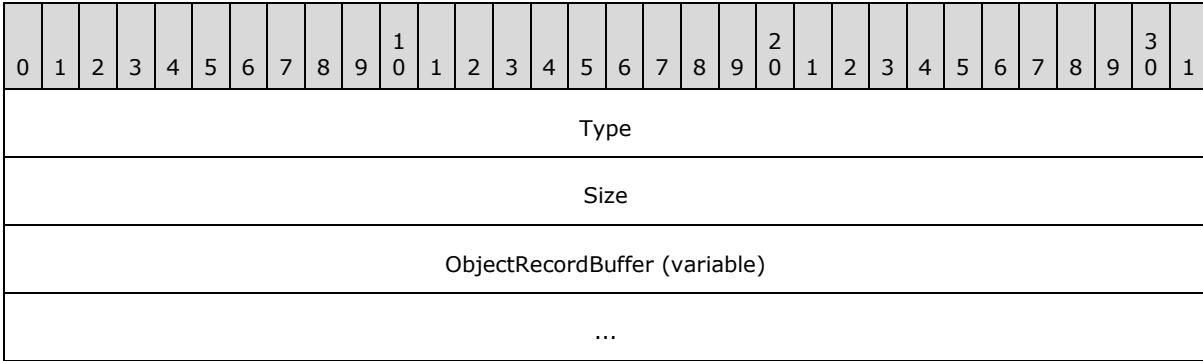
2.3.8 Object Manipulation Record Types

The object manipulation record types manage and modify graphics objects.

The following are EMF object manipulation record types.

Name	Section	Description
EMR_COLORCORRECTPALETTE	2.3.8.1	Specifies how to correct the entries of a LogPalette (section 2.2.17) object, using the WCS 1.0 members in the playback device context.
EMR_DELETECOLORSPACE	2.3.8.2	Specifies how to delete a logical color space from the EMF Object Table (section 3.1.1.1) .
EMR_DELETEOBJECT	2.3.8.3	Specifies the index of the object to be deleted from the EMF Object Table.
EMR_RESIZEPALETTE	2.3.8.4	Increases or decreases the size of an existing LogPalette object.
EMR_SELECTOBJECT	2.3.8.5	Specifies an existing object based on its index in the EMF Object Table.
EMR_SELECTPALETTE	2.3.8.6	Selects the specified LogPalette object into the playback device context.
EMR_SETCOLORSPACE	2.3.8.7	Specifies a logical color space, based on its index in the EMF Object Table.
EMR_SETPALETTEENTRIES	2.3.8.8	Defines RGB color values in a range of entries for an existing LogPalette object.

The generic structure of object manipulation records is specified as follows.



Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The object manipulation record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_SELECTOBJECT	0x00000025
EMR_DELETEOBJECT	0x00000028

Name	Value
EMR_SELECTPALETTE	0x00000030
EMR_SETPALETTEENTRIES	0x00000032
EMR_RESIZEPALETTE	0x00000033
EMR_SETCOLORSPACE	0x00000064
EMR_DELETECOLORSPACE	0x00000065
EMR_COLORCORRECTPALETTE	0x0000006F

Size (4 bytes): A 32-bit unsigned integer that specifies the size in bytes of this record in the metafile. This value MUST be a multiple of 4 bytes.

ObjectRecordBuffer (variable): An array of bytes that contains the remainder of the object manipulation record. The size of this field MUST be a multiple of 4 bytes.

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
ObjectRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

ObjectRecordParm (variable): An array of bytes that contains the parameters for the object manipulation record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.8.1 EMR_COLORCORRECTPALETTE Record

The EMR_COLORCORRECTPALETTE record specifies how to correct the entries of a logical palette object using WCS 1.0 values.[<91>](#)

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
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. This MUST be 0x00000018.

ihPalette (4 bytes): A 32-bit unsigned integer that specifies index of a logical palette object (section [2.2.17](#)) in the [EMF Object Table \(section 3.1.1.1\)](#).

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 EMR_DELETECOLORSPACE record deletes a logical color space object.[<92>](#)

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
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 index of a logical color space object in the [EMF Object Table \(section 3.1.1.1\)](#).

This object is either a WMF [LogColorSpace](#) or [LogColorSpaceW](#) object ([\[MS-WMF\]](#) sections [2.2.2.11](#) and [2.2.2.12](#), respectively).

An [EMR_DELETEOBJECT](#) record SHOULD be used instead of EMR_DELETECOLORSPACE to delete a logical color space object.[<93>](#)

See section [2.3.8](#) for additional object manipulation record types.

2.3.8.3 EMR_DELETEOBJECT Record

The 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	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
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 [EMR_RESIZEPALETTE](#) record increases or decreases the size of an existing [LogPalette](#) object (section [2.2.17](#)).

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
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 index of the palette object in the [EMF Object Table \(section 3.1.1.1\)](#).

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. [<94>](#)

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 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	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
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 EMF metafile 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 EMR_SELECTPALETTE record specifies a logical palette for the playback device context.

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
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. This MUST be 0x0000000C.

ihPal (4 bytes): A 32-bit unsigned integer that specifies either the index of a [LogPalette](#) object (section [2.2.17](#)) in the [EMF Object Table](#) or the value **DEFAULT_PALETTE**, which is the index of a stock object palette from the [StockObject](#) enumeration (section [2.1.31](#)).

This value MUST NOT be zero or 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 EMR_SETCOLORSPACE record defines the current logical color space object for graphics operations. [<95>](#)

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
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. This MUST be 0x0000000C.

ihCS (4 bytes): A 32-bit unsigned integer that specifies the index of a logical color space object in the [EMF Object Table \(section 3.1.1.1\)](#).

This object is either a WMF [LogColorSpace](#) or [LogColorSpaceW](#) object ([MS-WMF] sections [2.2.2.11](#) and [2.2.2.12](#), respectively).

The logical color space object defined by this record MUST be used in drawing operations that are specified by subsequent EMF records, until either a different logical color space object is specified by another EMR_SETCOLORSPACE record, or the object is removed by a [EMR_DELETECOLORSPACE](#) record.

See section [2.3.8](#) for additional object manipulation record types.

2.3.8.8 EMR_SETPALETTEENTRIES Record

The EMR_SETPALETTEENTRIES record defines RGB color values in a range of entries for an existing [LogPalette \(section 2.2.17\)](#) object.

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
Type																																		
Size																																		
ihPal																																		
Start																																		
NumberofEntries																																		
aPalEntries (variable)																																		
...																																		

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 (variable): An array of [LogPaletteEntry \(section 2.2.18\)](#) objects, of **NumberOfEntries** length, which 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.

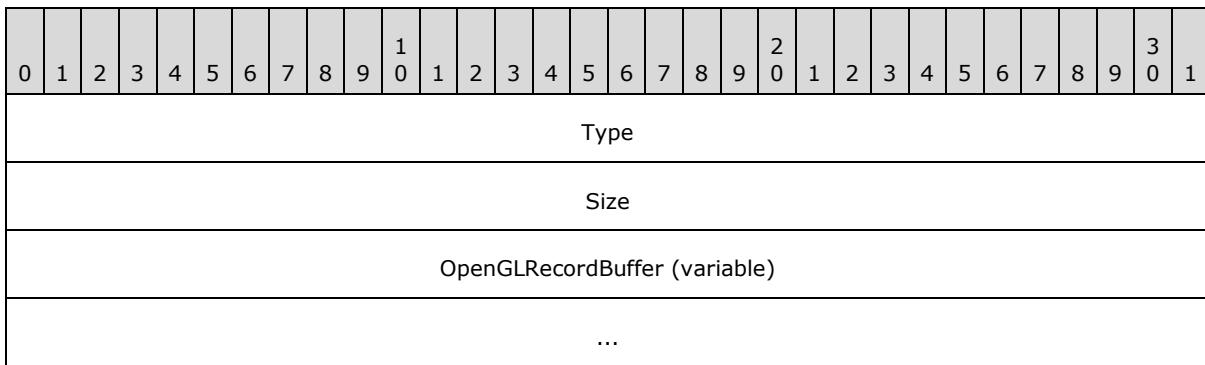
2.3.9 OpenGL Record Types

The OpenGL record types specify OpenGL functions.

The following are EMF OpenGL record types. [<96>](#)

Name	Section	Description
EMR_GLSBOUNDEDRECORD	2.3.9.1	Specifies an OpenGL function with a bounding rectangle for output.
EMR_GLSRECORD	2.3.9.2	Specifies an OpenGL function.

The generic structure of OpenGL records is specified as follows.

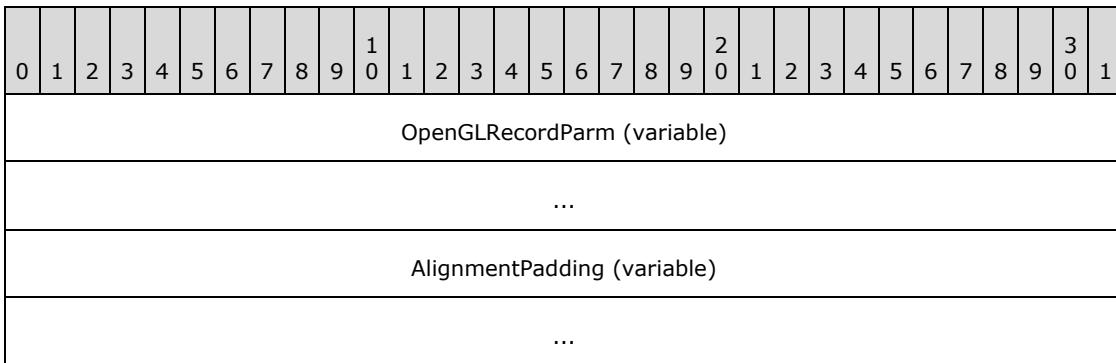


Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The OpenGL record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_GLSRECORD	0x00000066
EMR_GLSBOUNDEDRECORD	0x00000067

Size (4 bytes): A 32-bit unsigned integer that specifies the size in bytes of this record in the metafile. This value MUST be a multiple of 4 bytes.

OpenGLRecordBuffer (variable): An array of bytes that contains the remainder of the OpenGL record. The size of this field MUST be a multiple of 4 bytes.



OpenGLRecordParm (variable): An array of bytes that contains the parameters for the OpenGL record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See [\[OPENGL\]](#) for information about OpenGL functionality.

See section [2.3](#) for additional EMF record types.

2.3.9.1 EMR_GLSBOUNDEDRECORD Record

The EMR_GLSBOUNDEDRECORD record specifies an OpenGL function with a bounding rectangle for output.[<97>](#)

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
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.

Bounds (16 bytes): A WMF [RectL object](#) ([\[MS-WMF\]](#) section 2.2.2.19) that defines a bounding rectangle, in device units, for output produced by executing the OpenGL function.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the **Data** field. If this value is zero, no data is attached to this record.

Data (variable): An optional array of bytes of **cbData** length that specifies data for the OpenGL function.

See section [2.3.9](#) for additional OpenGL record types.

2.3.9.2 EMR_GLSRECORD Record

The EMR_GLSRECORD record specifies an OpenGL function.[<98>](#)

Note Fields that are not described in this section are specified in section [2.3.1](#).

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
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.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size, in bytes, of the **Data** field. If this value is zero, no data is attached to this record.

Data (variable): An optional array of bytes of **cbData** length that specifies data for the OpenGL function.

See section [2.3.9](#) for additional OpenGL record types.

2.3.10 Path Bracket Record Types

The path bracket record types specify and manipulate paths in path brackets.

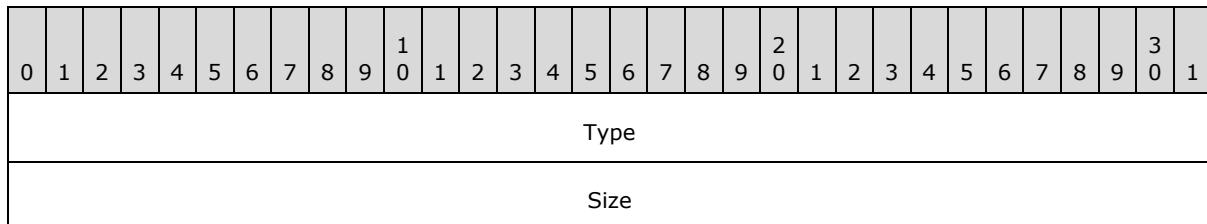
Note None of the path bracket records specify parameters.

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	This record opens a path bracket in the current playback device context. 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 EMR_ENDPATH record. When an application processes the EMR_BEGINPATH record, all previous paths MUST be discarded from the playback device context.
EMR_CLOSEFIGURE	This record closes an open figure in a path. Processing the EMR_CLOSEFIGURE 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 EMR_LINETO record instead of the EMR_CLOSEFIGURE record, end caps are used to create the corner instead of a join. EMR_LINETO is specified in section 2.3.5.13 . The EMR_CLOSEFIGURE record SHOULD only be used if there is an open path bracket in the playback device context. A figure in a path is open unless it is explicitly closed by processing this record.

Name	Description
	Note: A figure can be open even if the current point and the starting point of the figure are the same. After processing the EMR_CLOSEFIGURE record, adding a line or curve to the path MUST start a new figure.
EMR_ENDPATH	This record closes a path bracket and selects the path defined by the bracket into the playback device context.
EMR_FLATTENPATH	This record transforms any curves in the selected path into the playback device context; each curve MUST 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 drawn using the pen currently selected into the playback device context.

The generic structure of path bracket records is specified as follows.



Type (4 bytes): 32-bit unsigned integer that defines the type of the record. The types of records that specify no parameters are listed in the following table. See the preceding table 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 this record in the metafile. For path bracket records, this value MUST be 0x00000008

See section [2.3](#) for additional EMF record types.

2.3.11 State Record Types

The state record types specify and manage graphics properties that define the state of the playback device context.

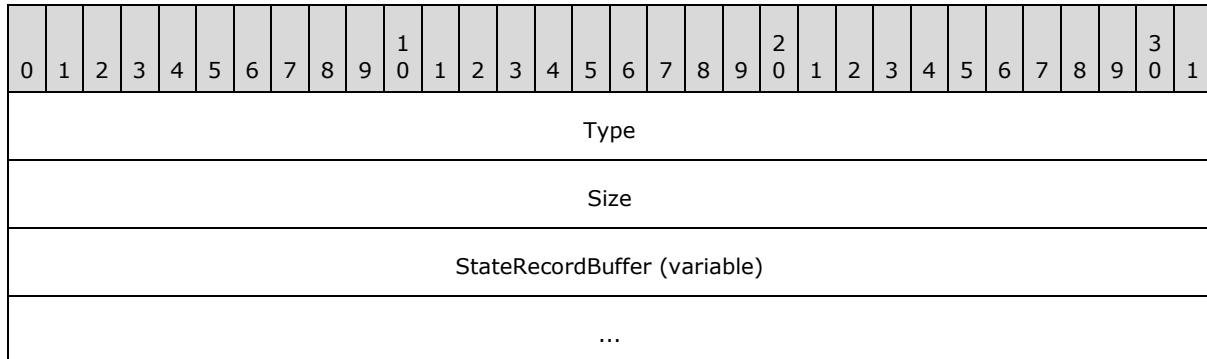
Note The EMR_REALIZEPALETTE and EMR_SAVEDC records do not specify parameters.

The following are EMF state record types.

Name	Section	Description
EMR_COLORMATCHTOTARGETW	2.3.11.1	Specifies how to preview colors as they would appear on the target device.
EMR_FORCEUFIMAPPING	2.3.11.2	Forces the font mapper to match fonts based on their UniversalFontId in preference to their LogFont (section 2.2.13) information.
EMR_INVERTRGN	2.3.11.3	Inverts the colors in the specified region.
EMR_MOVETOEX	2.3.11.4	Specifies the coordinates of a new drawing position, in logical units.
EMR_PIXELFORMAT	2.3.11.5	Specifies the pixel format.
EMR_REALIZEPALETTE	2.3.11	This record maps palette entries from the current LogPalette object (section 2.2.17) to the system_palette. This EMF record specifies no parameters.
EMR_RESTOREDC	2.3.11.6	Restores the playback device context to the specified state, which was saved by a preceding EMR_SAVEDC record.
EMR_SAVEDC	2.3.11	Saves the current state of playback device context on a stack of states saved by preceding EMR_SAVEDC records, if any. The state consists of graphics properties and objects, including the currently selected bitmap, brush, palette, font, pen, and region. An EMR_RESTOREDC record is used to restore the state. This EMF record specifies no parameters.
EMR_SCALEVIEWPORTEXTEX	2.3.11.7	Respecifies the viewport for the playback device context by using the ratios formed by the specified multiplicands and divisors.
EMR_SCALEWINDOWEXTEX	2.3.11.8	Respecifies the window for the playback device context by using the ratios formed by the specified multiplicands and divisors.
EMR_SETARCDIRECTION	2.3.11.9	Specifies the drawing direction to be used for arc and rectangle output.
EMR_SETBKCOLOR	2.3.11.10	Specifies the background color.
EMR_SETBKMODE	2.3.11.11	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_SETBRUSHORGEX	2.3.11.12	Specifies the origin of the current brush.
EMR_SETCOLORADJUSTMENT	2.3.11.13	Specifies color adjustment values to use in bitmap stretching.
EMR_SETICMMODE	2.3.11.14	Specifies ICM to be enabled, disabled, or queried on the playback device context.

Name	Section	Description
EMR_SETICMPROFILEA	2.3.11.15	Specifies how to set a specified color profile as the output profile for the playback device context.
EMR_SETICMPROFILEW	2.3.11.16	Specifies how to set a specified color profile as the output profile for the playback device context.
EMR_SETLAYOUT	2.3.11.17	Respecifies the layout of the playback device context.
EMR_SETLINKEDUFIS	2.3.11.18	Sets the UniversalFontIds (section 2.2.27) of the linked fonts to use during character lookup.
EMR_SETMAPMODE	2.3.11.19	Specifies the mapping mode for the playback device context.
EMR_SETMAPPERFLAGS	2.3.11.20	Respecifies the algorithm the font mapper uses when it maps logical fonts to physical fonts.
EMR_SETMITERLIMIT	2.3.11.21	Specifies the limit for the length of miter joins for the playback device context.
EMR_SETPOLYFILLMODE	2.3.11.22	Defines polygon fill mode.
EMR_SETROP2	2.3.11.23	Defines a binary raster operation mode.
EMR_SETSTRETCHBLTMODE	2.3.11.24	Specifies bitmap stretch mode.
EMR_SETTEXTALIGN	2.3.11.25	Specifies text alignment.
EMR_SETTEXTCOLOR	2.3.11.26	Defines the current text color.
EMR_SETTEXTJUSTIFICATION	2.3.11.27	Sets the amount of extra space to add to break characters for justification purposes.
EMR_SETVIEWPORTEXTEX	2.3.11.28	Defines the viewport extent.
EMR_SETVIEWPORTORGEX	2.3.11.29	Defines the viewport origin.
EMR_SETWINDOWEXTEX	2.3.11.30	Defines the window extent.
EMR_SETWINDOWORGEX	2.3.11.31	Defines the window origin.

The generic structure of state records is specified as follows.



Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The state record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_SETWINDOWORGEX	0x0000000A
EMR_SETVIEWPORTEXTEX	0x0000000B
EMR_SETVIEWPORTORGEX	0x0000000C
EMR_SETBRUSHORGEX	0x0000000D
EMR_SETCOLORADJUSTMENT	0x00000017
EMR_MOVETOEX	0x0000001B
EMR_SCALEVIEWPORTEXTEX	0x0000001F
EMR_SETWINDOWEXTEX	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_SCALEWINDOWEXTEX	0x00000020
EMR_SAVEDC	0x00000021
EMR_RESTOREDC	0x00000022
EMR_REALIZEPALETTE	0x00000034
EMR_SETARCDIRECTION	0x00000039
EMR_SETMITERLIMIT	0x0000003A
EMR_INVERTRGN	0x00000049
EMR_SETICMMODE	0x00000062
EMR_PIXELFORMAT	0x00000068
EMR_FORCEUFIMAPPING	0x0000006D

Name	Value
EMR_SETICMPFILEA	0x00000070
EMR_SETICMPFILEW	0x00000071
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 this record in the metafile. This value MUST be a multiple of 4 bytes.

StateRecordBuffer (variable): An optional array of bytes that contains the remainder of the state record. The size of this field MUST be a multiple of 4 bytes.

Note The EMR_REALIZEPALETTE and EMR_SAVEDC records do not contain this field.

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
StateRecordParm (variable)																																		
...																																		
AlignmentPadding (variable)																																		
...																																		

StateRecordParm (variable): An optional array of bytes that contains the parameters for the state record.

AlignmentPadding (variable): An optional array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section [2.3](#) for additional EMF record types.

2.3.11.1 EMR_COLORMATCHTOTARGETW Record

The EMR_COLORMATCHTOTARGETW record specifies whether to perform color matching with a color profile that is specified in a file with a name consisting of Unicode characters. [<99>](#)

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
Type																																		
Size																																		

dwAction
dwFlags
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](#) enumeration (section [2.1.7](#)).

dwFlags (4 bytes): A 32-bit unsigned integer that specifies a value from the [ColorMatchToTarget](#) enumeration (section [2.1.6](#)).

cbName (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in the Unicode UTF16-LE name of the desired color profile.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size of the raw data of the target color profile, if it is contained in the **Data** field.

Data (variable): An array of size (**cbName** + **cbData**) in bytes, which specifies the UTF16-LE name and raw data of the desired color profile.

An **EMR_COLORMATCHTOTARGETW** record can be used to control whether to apply the current color transform in the playback device context. If the **dwAction** value is **CS_ENABLE**, color mapping is enabled, and the current color transform SHOULD be applied to subsequent graphics operations. If **dwAction** is set to **CS_DISABLE**, the color transform SHOULD NOT be applied. [<100>](#)

While color mapping to the target is enabled by a **dwAction** value of **CS_ENABLE**, changes to the color space or color gamut mapping are not applied. 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 with an [EMR_SETICMMODE](#) record (section [2.3.11.14](#)).

See section [2.3.11](#) for additional state record types.

2.3.11.2 EMR_FORCEUFIMAPPING Record

The EMR_FORCEUFIMAPPING record forces the font mapper to match fonts based on their **UniversalFontId** in preference to their [LogFont](#) (section [2.2.13](#)) information.

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
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 \(section 2.2.27\)](#).

See section [2.3.5](#) for the specification of other Drawing record types.

2.3.11.3 EMR_INVERTRGN Record

The EMR_INVERTRGN record inverts the colors in the specified region.

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
Type																																		
Size																																		
Bounds																																		
...																																		
...																																		
RgnDataSize																																		
RgnData (variable)																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_INVERTRGN**. 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 WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.2.19, 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 EMR_MOVETOEX record specifies coordinates of the new current position, in logical units.

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
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 WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, which 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 EMR_PIXELFORMAT record specifies the pixel format to use for graphics operations. [<101>](#)

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
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](#) object (section [2.2.22](#)) that specifies pixel format data.

See section [2.3.11](#) for additional state record types.

2.3.11.6 EMR_RESTOREDC Record

The 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 that was created by prior [EMR_SAVEDC records \(section 2.3.11\)](#).

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
Type																																		
Size																																		
SavedDC																																		

Type (4 bytes): A 32-bit unsigned integer that identifies the record type as **EMR_RESTOREDC**. This value MUST be 0x00000022.

Size (4 bytes): A 32-bit unsigned integer that specifies the size of the record, in bytes. This value MUST be 0x0000000C.

SavedDC (4 bytes): A 32-bit signed integer that specifies the saved state to restore relative to the current state. This value MUST be negative; -1 represents the state that was most recently saved on the stack, -2 the one before that, etc.

The stack can contain state information for multiple instances of the playback device context. When a state is restored, all state instances that were saved more recently MUST be discarded.

See section [2.3.11](#) for the specification of other State record types.

2.3.11.7 EMR_SCALEVIEWPORTEXTEX Record

The 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	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
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 0x00000001F.

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.

```
xNewWE = (xOldWE * xNum) / xDenom
yNewWE = (yOldWE * yNum) / yDenom
```

See section [2.3.11](#) for the specification of other State record types.

2.3.11.8 EMR_SCALEWINDOWEXTEX Record

The 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	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
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.

```
xNewWE = (xOldWE * xNum) / xDenom
yNewWE = (yOldWE * yNum) / yDenom
```

See section [2.3.11](#) for the specification of other State record types.

2.3.11.9 EMR_SETARCDIRECTION Record

The 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	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
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. This value MUST be 0x0000000C.

ArcDirection (4 bytes): A 32-bit unsigned integer that specifies the arc direction. The value MUST be in the [ArcDirection](#) enumeration (section [2.1.2](#)).

The default direction is counterclockwise.

The **EMR_SETARCDIRECTION** record affects the direction in which the following records draw:

- [EMR_ARC \(section 2.3.5.2\)](#)
- [EMR_ARCTO \(section 2.3.5.3\)](#)
- [EMR_CHORD \(section 2.3.5.4\)](#)
- [EMR_ELLIPSE \(section 2.3.5.5\)](#)
- [EMR_PIE \(section 2.3.5.15\)](#)
- [EMR_RECTANGLE \(section 2.3.5.34\)](#)
- [EMR_ROUNDRECT \(section 2.3.5.35\)](#)

See section [2.3.11](#) for the specification of other State record types.

2.3.11.10 EMR_SETBKCOLOR Record

The EMR_SETBKCOLOR record specifies the background color.

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
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 WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.2.8, which 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 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	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
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 EMR_SETBRUSHORGEX record specifies the origin of the current brush.

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
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 WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.2.15, which specifies the brush's 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 EMR_SETCOLORADJUSTMENT record specifies color adjustment properties in the playback device context.

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
Type																																		
Size																																		
ColorAdjustment																																		
...																																		
...																																		
...																																		
...																																		
...																																		
...																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_SETCOLORADJUSTMENT**. This MUST be 0x00000017.

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

ColorAdjustment (24 bytes): A [ColorAdjustment object \(section 2.2.2\)](#) that specifies color adjustment values.

Color adjustment values are used to adjust the input color of the source bitmap for graphics operations performed by [EMR_STRETCHBLT](#) and [EMR_STRETCHDIBITS](#) records when **STRETCH_HALFTONE** mode is set from the [StretchMode](#) enumeration (section 2.1.32).

The ColorAdjustment object specified by this record MUST be used in graphics operations that require a ColorAdjustment object, until a different ColorAdjustment object is specified by another EMR_SETCOLORADJUSTMENT record, or until 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 EMR_SETICMMODE record specifies the mode of Image Color Management (ICM) for graphics operations. [<102>](#)

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
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. This MUST be 0x0000000C.

ICMMode (4 bytes): A 32-bit unsigned integer that specifies whether to enable or disable ICM, from the [ICMMode](#) enumeration (section [2.1.18](#)). This value is part of the state of the playback device context.

When ICM mode is enabled, colors specified in EMF records SHOULD be color matched, whereas the default color profile in the playback device context SHOULD be used when a bit-block transfer is performed. If the default color profile is not desired, ICM mode SHOULD be turned off before performing the bit-block transfer.

See section [2.3.11](#) for additional state record types.

2.3.11.15 EMR_SETICMPROFILEA Record

The EMR_SETICMPROFILEA record specifies a color profile in a file with a name consisting of ASCII characters, for graphics output. [<103>](#)

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
Type																																		
Size																																		
dwFlags																																		
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.

dwFlags (4 bytes): A 32-bit unsigned integer that contains color profile flags.

cbName (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in the ASCII name of the desired color profile.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size of the color profile data, if it is contained in the **Data** field.

Data (variable): An array of size (**cbName** + **cbData**) in bytes, which specifies the ASCII name and raw data of the desired color profile.

See section [2.3.11](#) for additional state record types.

2.3.11.16 EMR_SETICMPROFILEW Record

The EMR_SETICMPROFILEW record specifies a color profile in a file with a name consisting of Unicode characters, for graphics output.[<104>](#)

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
Type																																		
Size																																		
dwFlags																																		
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.

dwFlags (4 bytes): A 32-bit unsigned integer that contains color profile flags.

cbName (4 bytes): A 32-bit unsigned integer that specifies the number of bytes in the Unicode UTF16-LE name of the desired color profile.

cbData (4 bytes): A 32-bit unsigned integer that specifies the size of color profile data, if attached.

Data (variable): An array of size (**cbName** + **cbData**) in bytes, which specifies the UTF16-LE name and raw data of the desired color profile.

See section [2.3.11](#) for additional state record types.

2.3.11.17 EMR_SETLAYOUT Record

The EMR_SETLAYOUT record specifies the order in which text and graphics are drawn. [<105>](#)

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
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. This MUST be 0x0000000C.

LayoutMode (4 bytes): A 32-bit unsigned integer that specifies the layout mode as follows:

Value	Meaning
LAYOUT_LTR 0x00000000	Sets the default horizontal layout to be left-to-right. This is the default mode for English and European locales.
LAYOUT_RTL 0x00000001	Sets the default horizontal layout to be right-to-left. This mode is required for some languages, including Arabic and Hebrew.
LAYOUT_BITMAPORIENTATIONPRESERVED 0x00000008	Disables mirroring of bitmaps that are drawn by Bitmap Record Types (section 2.3.1) , when the layout mode is right-to-left.

See section [2.3.11](#) for additional state record types.

2.3.11.18 EMR_SETLINKEDUFI Record

The EMR_SETLINKEDUFI record sets the [UniversalFontIds \(section 2.2.27\)](#) of the linked fonts to use during character lookup.

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
Type																																		
Size																																		
uNumLinkedUFI																																		
ufis (variable)																																		

...
Reserved
...

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_SETLINKEDUFI**. 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 UFIs to follow.

ufis (variable): An array of **uNumLinkedUFI** elements of type UniversalFontId, which specifies 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 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-axis and y-axis.

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
Type																																		
Size																																		
MapMode																																		

Type (4 bytes): A 32-bit unsigned integer that identifies this record type as **EMR_SETMAPMODE**. This MUST be 0x00000011.

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](#) enumeration (section [2.1.21](#)).

MM_TEXT mode allows applications to work in device pixels, whose size varies from device to device.

The **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 EMR_SETMAPPERFLAGS record specifies parameters of the process of matching logical fonts to physical fonts, which is performed by the font mapper. [<106>](#)

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
Type																																		
Size																																		
Flags																																		

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. This MUST be 0x0000000C.

Flags (4 bytes): A 32-bit unsigned integer that specifies parameters of the font matching process.

Value	Meaning
0x00000001	The font mapper SHOULD select only fonts that match the aspect ratio of the output device, as it is currently defined in the playback device context.

See section [2.3.11](#) for additional state record types.

2.3.11.21 EMR_SETMITERLIMIT Record

The 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	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
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 unsigned integer that specifies the new miter length limit. [<107>](#)

See section [2.3.11](#) for the specification of other State record types.

2.3.11.22 EMR_SETPOLYFILLMODE Record

The EMR_SETPOLYFILLMODE record defines polygon fill mode.

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
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 nonzero 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 EMR_SETROP2 record defines a binary raster operation mode.

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
Type																																		
Size																																		

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 WMF [Binary Raster Op](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.2).

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 EMR_SETSTRETCHBLTMODE record specifies bitmap stretch mode.

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
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 that the [EMR_STRETCHBLT](#) record is processed on.

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_SETBRUSHORGEX](#) 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 EMR_SETTEXTALIGN record specifies text alignment.

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
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 text alignment flags. These are either WMF [TextAlignmentMode Flags](#) ([\[MS-WMF\]](#) section 2.1.2.3) for text with a horizontal baseline, or WMF [VerticalTextAlignmentMode Flags](#) ([\[MS-WMF\]](#) section 2.1.2.4) for text with a vertical baseline. Only one value can be chosen from those that affect horizontal and vertical alignment.

The [EMR_SMALLTEXTOUT](#), [EMR_EXTEXTOUTA](#), and [EMR_EXTEXTOUTW](#) 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 EMR_SETTEXTCOLOR record defines the current text color.

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
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 WMF [ColorRef](#) object ([\[MS-WMF\]](#) section 2.2.2.8) 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 EMR_SETTEXTJUSTIFICATION record specifies the amount of extra space to add to break characters for text justification. [<108>](#)

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
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.

Instead of using an EMR_SETTEXTJUSTIFICATION record, an implementation SHOULD use an [EMR_EXTTEXTOUTW](#) record (section [2.3.5.8](#)) to perform this function. [<109>](#)

See section [2.3.11](#) for the specification of other state record types.

2.3.11.28 EMR_SETVIEWPORTEXTEX Record

The EMR_SETVIEWPORTEXTEX record defines the viewport extent.

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
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 WMF [SizeL](#) object ([\[MS-WMF\]](#) section 2.2.2.22) 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 EMR_SETVIEWPORTORGEX record defines the viewport origin.

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
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 WMF [PointL](#) object ([\[MS-WMF\]](#) section 2.2.2.15) 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 EMR_SETWINDOWEXTEX record defines the window extent.

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
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 WMF [SizeL](#) object ([\[MS-WMF\]](#) section 2.2.2.22) 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 EMR_SETWINDOWORGEX record defines the window origin.

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
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 WMF [PointL](#) object ([\[MS-WMF\]](#) section 2.2.2.15) 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 transform record types specify and modify world-space to page-space transforms.

The following are EMF transform record types.

Name	Section	Description
EMR_MODIFYWORLDTRANSFORM	2.3.12.1	Modifies the current world-space to page-space transform.
EMR_SETWORLDTRANSFORM	2.3.12.2	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	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
Type																																		

Size
Xform
...
...
...
...
...
TransformData (optional)

Type (4 bytes): A 32-bit unsigned integer that defines the type of record. The transform record types are listed in the following table. See the preceding table 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 this record in the metafile. This value MUST be a multiple of 4 bytes.

Xform (24 bytes): An [XForm object \(section 2.2.28\)](#), which defines a world-space to page-space transform.

TransformData (4 bytes): An optional 32-bit unsigned integer that specifies an additional parameter for the record.

See section [2.3](#) for additional EMF record types.

2.3.12.1 EMR MODIFYWORLDTRANSFORM Record

The EMR MODIFYWORLDTRANSFORM record modifies the current world-space to page-space transform in the playback device context.

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
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. This MUST be 0x00000024.

Xform (24 bytes): An [XForm object \(section 2.2.28\)](#) that defines a two-dimensional linear transform in logical units. This transform is used according to the **ModifyWorldTransformMode** to define a new value for the world-space to page-space transform in the playback device context.

ModifyWorldTransformMode (4 bytes): A 32-bit unsigned integer that specifies how the transform specified in **Xform** is used. This 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 EMR_SETWORLDTRANSFORM record specifies a transform for the current world-space to page-space transform in the playback device context.

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
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. This MUST be 0x00000020.

Xform (24 bytes): An [XForm object \(section 2.2.28\)](#) that specifies a two-dimensional linear transform in logical units. This transform defines a new value for the world-space to page-space transform in the playback device context.

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

3.1 Metafile Design

3.1.1 Managing Objects

3.1.1.1 EMF Object Table

When a graphics object is created by an EMF object creation record (section [2.3.7](#)), the record specifies a numerical index. The object can be referenced by its index during metafile processing until the object is deleted. Object indexes start at 1; zero is reserved for references to the metafile itself.

An object manipulation record (section [2.3.8](#)) can use the index of a graphics object to select it into the playback device context. This has the effect of activating the object so it can be used in graphics operations specified by subsequent metafile records. Until the object is activated, it is not used. Later, if a different object of the same type is activated, the former object is deactivated but not deleted. An object is not deleted until an object manipulation record is processed that deletes it.

Before a graphics object of a particular type is created and activated, a stock object (section [2.1.31](#)) of that type is used in graphics operations.

The EMF object table refers to an abstract model for managing graphics objects during the processing of an EMF metafile. It consists of a table in which graphics objects are associated with their indexes as they are created, activated, used, deactivated, and deleted.

The following types of graphics objects are managed in this model:

- Brushes
- Color spaces
- Fonts
- Palettes
- Pens

The following process model can be used to manage graphics objects with an EMF object table:

1. At the start of metafile processing, an EMF object table is created. The maximum number of objects is specified in the EMF [Header](#) object (section [2.2.9](#)) of the [EmfMetafileHeader](#) record (section [2.3.4.2.1](#)).
2. The EMF object table needs to be large enough to keep track of objects that are explicitly created as well as stock objects. Each element in the object table contains information that indicates whether an object with that index has been created, a way to access the object, and whether the object is currently active.
3. When a graphics object is created, the element in the EMF object table that corresponds to its index is updated so that it can be accessed later.
4. When a graphics object is activated, the element in the EMF object table that corresponds to its index is updated. In addition, the element that was activated before is now deactivated.

5. When a graphics object is deactivated, the element in the EMF object table that corresponds to its index is updated. In addition, the default stock object of that type is now activated.
6. When a record is encountered that deletes the graphics object, its memory is released and the EMF object table is updated accordingly. A graphics object can be deleted without first being deactivated. If that happens, the default stock object of that type is now activated.

Note There are some index values that are reserved:

- The index zero is reserved; it refers to the EMF metafile itself.
- Indexes that have the most-significant bit set refer to stock objects.

The object state changes of creation, activation, deactivation, and deletion, require management during playback to achieve the expected results in rendering the image stored in the metafile.

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 could 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.

3.2 EMF Metafile Example

This section describes an example of a metafile, which when processed, renders the following image.



Figure 3: EMF Metafile Example

The contents of this metafile example are shown in this section in hexadecimal bytes. The far-left column is the byte count; the far-right characters are the interpretation of the bytes in the Latin-1 **ANSI Character Set**, as specified in [ISO/IEC-8859-1](#). The sections that follow describe the packets that convey this series of bytes.


```

000035E0:20 AC CF 02 00 00 00 00 00 00 07 02 00 00 00 00 00 00 -í.....
000035F0:10 AC CF 02 00 00 00 00 26 36 E3 76 00 00 00 00 00 00 .-í....&6äv....
00003600:00 00 07 02 00 00 00 01 00 00 00 00 27 00 00 .....'..
00003610:00 00 00 00 00 00 00 20 AC CF 02 64 76 00 08 ..... -í.Dv..
00003620:00 00 00 25 00 00 00 0C 00 00 00 04 00 00 00 .....%.....
00003630:28 00 00 00 0C 00 00 03 00 00 00 52 00 00 00 00 (..... R...
00003640:70 01 00 00 03 00 00 13 00 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 00 N...N...É....
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 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 00 i.F.....
000036C0:FF FF 5A FE 00 00 00 00 40 02 5A FE FE 07 00 00 ýýzb...@.zpb...
000036D0:9D 04 00 00 00 00 00 FF FF FF FF FF FF FF •.....ýýýýýýýý
000036E0:01 00 00 00 00 00 00 20 AC CF 02 00 00 00 00 ..... -í.....
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 F3 14 0A 1E 00 00 00 00 00 ó.....ó.....
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00 "Šépb.....
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 .ó.....Vå%...
00003740:55 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 U.....
00003750:00 00 00 FE 07 00 00 79 0D 21 11 00 00 00 00 00 .....pb..y.!....
00003760:40 02 5A FE 00 00 00 26 06 5A FE FE 07 00 00 @.zpb...&.zpb...
00003770:08 F5 13 00 00 00 00 00 F5 13 00 00 00 00 00 00 00 .ó.....ó.....
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00 .ÉTpb..y.!....
00003790:04 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 00 ....Dv.....%...
000037B0:0C 00 00 00 03 00 00 00 25 00 00 00 0C 00 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 03 00 00 00 25 00 00 00 00 (.....%...
000037E0:0C 00 00 00 0D 00 00 80 0E 00 00 00 14 00 00 00 .....€.....
000037F0:00 00 00 10 00 00 00 14 00 00 00 ..... .

```

3.2.1 EMR_HEADER Example

This section provides an example of the [EMR_HEADER](#) record (section [2.3.4.2](#)).

```

00000000:01 00 00 00 D4 00 00 00 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 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 00 00
00000040:6C 00 00 00 00 00 00 80 07 00 00 B0 04 00 00 00 00
00000050:A5 02 00 00 A7 01 00 00 00 00 00 00 00 00 00 00 00 00
00000060:00 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	1	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 (0x000000000)																																
... (0x000000000)																																
... (0x000000059)																																
... (0x000000059)																																
Frame (0x000000000)																																
... (0x000000000)																																
... (0x00000C42)																																
... (0x00000C31)																																

Figure 4: EMR_HEADER record example, part 1

Type: 0x00000001 identifies the record type as EMR_HEADER.

Size: 0x000000D4 is the record size, in bytes.

Bounds: 0x000000000, 0x000000000, 0x00000059, 0x00000059 specify the rectangular inclusive-inclusive bounds, in device units, of the smallest rectangle that can be drawn around the image stored in the metafile.

Frame: 0x000000000, 0x000000000, 0x00000C42, 0x00000C31 specify 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	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1															
Signature (0x464D4520)																																															
Version (0x00010000)																																															
Byte (0x000037FC)																																															
Records (0x00000016)																																															
Handles (0x0005)																Reserved (0x0000)																															
nDescription (0x00000034)																																															
offDescription (0x0000006C)																																															
PalEntries (0x000000000)																																															

Figure 5: EMR_HEADER record example, part 2

Signature: 0x464D4520 is the record signature, which consists of the ASCII string "EMF".

Version: 0x00010000 specifies EMF metafile interoperability.

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 graphics objects that will be created during the processing of the metafile. These objects are referenced by their indexes in metafile records. Index values for created objects start at 1. This value can be used to compute the size needed for the [EMF Object Table \(section 3.1.1.1\)](#).

Reserved: 0x0000 is ignored.

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	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
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 6: EMR_HEADER record example, part 3

Device: 0x00000780, 0x00000780 specify the size of the reference device, in pixels.

Millimeters: 0x000002A5, 0x000001A7 specify the size of the reference device, in millimeters.

cbPixelFormat: 0x00000000 specifies the size of the [PixelFormatDescriptor \(section 2.2.22\)](#) 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 specify 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 [EMR_CREATEBRUSHINDIRECT](#) record (section [2.3.7.1](#)).

```
0000000D0: 27 00 00 00 18 00 00 00 01 00 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	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
Type (0x00000027)																																		
Size (0x00000018)																																		
ihBrush (0x00000001)																																		
LogBrush (0x00000002)																																		
... (0x0052472A)																																		
... (0x00000003)																																		

Figure 7: EMR_CREATEBRUSHINDIRECT record example, part 1

Type: 0x00000027 identifies this record type as EMR_CREATEBRUSHINDIRECT.

Size: 0x00000018 specifies the size of this record, in bytes.

ihBrush: 0x00000001 specifies the index of this brush object in the [EMF Object Table](#) (section [3.1.1.1](#)).

LogBrush: A [LogBrushEx](#) object (section [2.2.12](#)) that contains brush data.

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
BrushStyle (0x00000002)																																		
Color (0x0052472A)																																		
BrushHatch (0x00000003)																																		

Figure 8: EMR_CREATEBRUSHINDIRECT record example, part 2

BrushStyle: 0x00000002 specifies a hatch brush style, from the WMF **BrushStyle** enumeration ([\[MS-WMF\]](#) section [2.1.1.4](#)).

Color: 0x0052472A is a WMF **ColorRef** object ([\[MS-WMF\]](#) section [2.2.2.8](#)) that specifies the brush color value.

BrushHatch: 0x00000003 specifies the brush hatch. Its interpretation depends on the value of **BrushStyle**. In this case, it specifies a 45-degree upward, left-to-right hatch pattern.

3.2.3 EMR_SELECTOBJECT Example

This section provides an example of the [EMR_SELECTOBJECT](#) record (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	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000001)																															

Figure 9: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_BITBLT](#) record (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 00 00 00 80 3F
00000130:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F
00000140:00 00 00 00 00 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 00 00 00 00 00 00

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
Type (0x0000004C)																																		
Size (0x00000064)																																		
Bounds (0x0000000000)																																		
... (0x0000000000)																																		
... (0x000000059)																																		
... (0x000000059)																																		
xDest (0x0000000000)																																		
yDest (0x0000000000)																																		
cxDest (0x000000059)																																		
cyDest (0x000000059)																																		

Figure 10: EMR_BITBLT record example, part 1

Type: 0x0000004C identifies this record type as EMR_BITBLT.

Size: 0x00000064 specifies the size of this record, in bytes.

Bounds: 0x00000000, 0x00000000, 0x00000059, 0x00000059 specify 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	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
BitBlitRasterOperation (0x00F00021)																																		
xSrc (0x0000000000)																																		
ySrc (0x0000000000)																																		
xformSrc (0x3F800000)																																		
... (0x0000000000)																																		
... (0x0000000000)																																		
... (0x3F800000)																																		
... (0x0000000000)																																		
... (0x0000000000)																																		

Figure 11: EMR_BITBLT record example, part 2

BitBlitRasterOperation: 0x00F00021 specifies the raster operation code from the WMF **Ternary Raster Operation** enumeration ([\[MS-WMF\]](#) section 2.1.1.31). This code defines 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.

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 specify the world-space to page-space transform. 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 (0x00000000)																															
UsageSrc (0x00000000)																															
offBmiSrc (0x00000000)																															
cbBmiSrc (0x00000000)																															
offBitsSrc (0x00000000)																															
cbBitsSrc (0x00000000)																															

Figure 12: EMR_BITBLT Record Example, Part 3

BkColorSrc: 0x00000000 specifies the background RGB color.

UsageSrc: 0x00000000 specifies the value of the **Colors** field of the WMF **DeviceIndependentBitmap** object ([\[MS-WMF\]](#) section 2.2.2.9) from the **DIBColors** enumeration (section [2.1.9](#)).

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 [EMR_SELECTOBJECT](#) record (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	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000025)																														
Size (0x0000000C)																														
ihObject (0x80000000 = WHITE BRUSH)																														

Figure 13: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_BITBLT record \(section 2.3.1.2\)](#).

```

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 00 00 80 3F
000001A0:00 00 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 00 64 00 00 00 00
000001C0:28 00 00 00 8C 00 00 00 1C 2F 00 00 28 00 00 00 00
000001D0:59 00 00 00 2D 00 00 00 01 00 18 00 00 00 00 00 00 00
000001E0:1C 2F 00 00 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 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 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 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 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 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
000003E0:0C 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
00000420:0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E
00000430:41 5E 0E 41
00000440:5E 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
000004E0:0C 43 5C 0C
000004F0:43 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 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
00000550:0E 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
000005F0:43 5C 0C 43
00000600:5C 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
00000650:0E 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 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
00000700:5C 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
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
00000800:5C 0C 43 5C
00000810:0C 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
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
00000910:0C 43 5C 0C
00000920:43 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
00000A20:43 5C 0C 43
00000A30:5C 0C 43 5C
00000A40:0C 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
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
00000B30:5C 0C 43 5C
00000B40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0A
00000B50:45 5C 0A 45 5B 0C 00
00000B60:3F 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
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
00000C30:5C 0C 43 5C
00000C40:0C 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
00000C80:0E 40 5D 0D 3E 5E 0D 3E 5E 0D 40 5D 0D 40 5D 0D
00000C90:40 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
00000D40:0C 43 5C 0C
00000D50:43 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
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
00000E60:5C 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
00000EB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E

00000E00:40 5D 0D 40 5D 0D 40 5D 0D 42 5C 0E 60 6E 33 A9
00000E00:B6 91 CD B0 8B D7 A2 78 D6 99 6D D9 A8 86 E2 D7
00000E00:CA DE BC A2 CC 78 44 D0 85 57 D2 8B 61 CC 74 42
00000E00: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
00000F70:0C 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
00000FA0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000FB0:0D 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
00001070:0C 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
000010B0:0E 40 5D 0D
000010C0:40 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
00001180:43 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
000011D0:5D 0D 40 5D
000011E0:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
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
00001290:5C 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
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
000013A0:0C 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
000013D0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000013E0:0D 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
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
000014F0:40 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
000015B0:43 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
00001600:5D 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
000016C0:5C 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
00001710:0D 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
000017D0:0C 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
00001820:40 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
000018E0:43 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
00001920:40 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
000019F0:5C 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
00001A30:5D 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
00001AF0:5C 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
00001B40:0D 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 C0 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
00001C00:0C 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
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
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
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
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
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
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
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
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
00002110:42 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
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
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
00002290:5D 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
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
00002390:5D 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
00002430:0B 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
00002490:5D 0D 40 5D
000024A0:0D 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

000024E0:F0 F0 F0 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 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
00002540:42 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
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
000025A0:0D 40 5D 0D
000025B0:40 5D 0D 40
000025C0:5D 0D 40 5D 0D 3F 5C 0C 52 64 1D A1 92 80 C5 C5
000025D0:C4 D2 D2 E1 E2 E3 EB EC ED F2 F2 F3 F5 F5 F5
000025E0:F7 F7 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
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
000026B0:40 5D 0D 40
000026C0:5D 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 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
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
000027C0:5D 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 F8 F6 F4 EE E9 E4 E5 DE D7 E6 E1
00002800:DB E4 E0 DB B8 AA 7D B7 AD 7C 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
00002860:0B 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
000028C0:5D 0D 40 5D
000028D0:0D 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
00002970:42 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
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
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
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
00002B80:5B 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
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
00002D20:5D 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
00002D70:41 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
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
00002DD0:43 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
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
00002EF0:09 42 5E 09 42 5E 09 00 3F 64 0B 3F 64 0B 3F 64
00002F00:0B 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
00002F70:40 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
00002FA0:40 61 0B 40
00002FB0:61 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	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
Type (0x0000004C)																															
Size (0x00002FA8)																															
Bounds (0x0000000000)																															
... (0x00000002D)																															
... (0x000000059)																															
... (0x000000059)																															
xDest (0x0000000000)																															
yDest (0x00000002D)																															
cxDest (0x00000005A)																															
cyDest (0x00000002D)																															

Figure 14: EMR_BITBLT record example, part 1

Type: 0x0000004C identifies this record type as EMR_BITBLT.

Size: 0x00002FA8 specifies the size of this record, in bytes.

Bounds: 0x0000000000, 0x00000002D, 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: 0x00000002D 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	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
BitBlitRasterOperation (0x00CC0020)																															
xSrc (0x00000000)																															
ySrc (0x00000000)																															
xformSrc (0x3F800000)																															
... (0x00000000)																															
... (0x00000000)																															
... (0x3F800000)																															
... (0x00000000)																															
... (0x00000000)																															

Figure 15: EMR_BITBLT record example, part 2

BitBlitRasterOperation: 0x00CC0020 specifies the raster operation code from the WMF **Ternary Raster Operation** enumeration ([MS-WMF](#) section 2.1.1.31). This code defines 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.

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 specify the world-space to page-space transform. For more information about coordinate spaces, see [MSDN-WRLDPGSPC](#).

0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
BkColorSrc (0x00FFFFFF)																															
UsageSrc (0x00000000)																															
offBmiSrc (0x00000064)																															
cbBmiSrc (0x00000000)																															
offBitsSrc (0x0000008C)																															
cbBitsSrc (0x00002F1C)																															

Figure 16: EMR_BITBLT record example, part 3

BkColorSrc: 0x00FFFFFF specifies the background RGB color.

UsageSrc: 0x00000000 specifies the value of the **Colors** field of the WMF **DeviceIndependentBitmap** object ([MS-WMF](#) section 2.2.2.9) from the **DIBColors** enumeration (section [2.1.9](#)).

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 [EMR_SETBKMODE](#) record (section [2.3.11.11](#)).

00003110:12 00 00 00 0C 00 00 00 01 00 00 00 00

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000012)																																
Size (0x0000000C)																																
Mode (0x00000001)																																

Figure 17: EMR_SETBKMODE record 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 [EMR_EXTCREATEFONTINDIRECTW](#) record (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 00
00003130:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 00 00 01
00003140:04 00 00 02 41 00 72 00 69 00 61 00 6C 00 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 07 00 00 20 00 CC 00 00 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 00
00003190:0E 20 05 27 00 00 00 00 28 00 00 00 00 00 00 00 00
000031A0:1C 2F 00 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 00 00
000031C0:00 00 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 00 00
000031F0:58 00 00 00 2C 00 00 00 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 00 00 00 80 3F
00003210:00 00 00 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 00 00
00003250:10 00 90 01 00 00 00 00 25 00 00 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

```

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	3	1
Type (0x00000052)																								
Size (0x000000170)																								
ihFonts (0x000000002)																								
elw (variable)																								
...																								
(elw cont'd for 87 rows)																								

Figure 18: EMR_EXTCREATEFONTINDIRECTW record example

Type: 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size: 0x000000170 specifies the size of this record, in bytes.

ihFonts: 0x000000002 specifies the object index that is assigned to the font.

elw: To determine the type of logical font object in this field, an algorithm (section [2.3.7.8](#)) is applied, which indicates this is a variable-length [LogFontExDv object \(section 2.2.15\)](#).

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	3	1
Height (0xFFFFFFF3)																								
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 19: LogFontExDv object, part 1

Height: 0xFFFFFFF3 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 WMF [CharacterSet](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.5).

OutPrecision: 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the WMF [OutPrecision](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.21). The value 0x04 specifies 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](#) flags ([\[MS-WMF\]](#) section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality: 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.10).

PitchAndFamily: 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [PitchAndFamily object](#) ([\[MS-WMF\]](#) section 2.2.2.14).

Facename: "Arial" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	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 20: 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: 0x08007664 specifies the signature of a [DesignVector object \(section 2.2.3\)](#).

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 [EMR_SELECTOBJECT](#) record (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	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000002)																															

Figure 21: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_EXTTEXTOUTW](#) record (section [2.3.5.8](#)).

```
00003290: 54 00 00 00 A0 00 00 00  
000032A0:00 00 00 00 00 00 00 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 00  
000032D0:00 00 00 00 00 00 00 FF FF 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	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
Type (0x00000054)																															
Size (0x000000A0)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0xFFFFFFFF)																															
... (0xFFFFFFFF)																															
iGraphicsMode (0x00000001)																															
exScale (35.260418)																															
eyScale (35.250000)																															
aemrtext (variable)																															

Figure 22: EMR_EXTTEXTOUTW record example

Type: 0x00000054 identifies this record type as EMR_EXTTEXTOUTW.

Size: 0x000000A0 specifies the size of this record, in bytes.

Bounds: 0x00000000, 0x00000000, 0xFFFFFFFF, 0xFFFFFFFF values are not used.

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](#) objects (section [2.2.5](#)) 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	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reference (0x000000012)																															
... (0x00000005)																															
Chars (0x0000000E)																															
offString (0x00000004C)																															
Options (0x000000000)																															
Rectangle (0x000000000)																															
... (0x000000000)																															
... (0xFFFFFFFF)																															
... (0xFFFFFFFF)																															
offDx (0x00000068)																															
text ("Simple Sample")																															

Figure 23: EmrText object example

Reference: 0x00000012, 0x00000005 specify the coordinates of the reference point used to position the string.

Chars: 0x0000000E specifies the number of characters in the string.

offString: 0x00000004C 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 [EMR_EXTCREATEFONTINDIRECTW](#) record (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 00
00003390:00 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 00 00 00 10 00 90 01
000033D0:00 00 00 25 00 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 59 00 00 00 2D 00 00 00 00
00003400:64 76 00 08 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 00 00
00003420:00 00 00 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 00 00 00
00003440:00 00 00 00 92 A0 CD 02 00 00 00 00 00 00 00 00 00 00 00 00
00003450:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003460:00 00 00 00 00 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 01 00 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	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
Type (0x00000052)																															
Size (0x000000170)																															
ihFonts (0x000000003)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

Figure 24: EMR_EXTCREATEFONTINDIRECTW record example

Type: 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size: 0x000000170 specifies the size of this record, in bytes.

ihFonts: 0x00000003 specifies the object index that is assigned to the font.

elw: To determine the type of logical font object in this field, an algorithm (section 2.3.7.8) is applied, which indicates this is a variable-length [LogFontExDv object \(section 2.2.15\)](#).

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
Height (0xFFFFFFF3)																																	
Width (0x00000000)																																	
Escapement (0x000000C4E)																																	
Orientation (0x000000C4E)																																	
Weight (0x0000000C8)																																	
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 25: LogFontExDv object, part 1

Height: 0xFFFFFFF3 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: 0x000000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

Orientation: 0x000000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

Weight: 0x0000000C8 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_CHARSET**, as defined in the WMF **CharacterSet** enumeration ([\[MS-WMF\]](#) section 2.1.1.5).

OutPrecision: 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the WMF **OutPrecision** enumeration ([\[MS-WMF\]](#) section 2.1.1.21). The value 0x04 specifies 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** flags ([\[MS-WMF\]](#) section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality: 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.10).

PitchAndFamily: 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [PitchAndFamily object](#) ([\[MS-WMF\]](#) section 2.2.2.14).

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 26: 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 a [DesignVector object \(section 2.2.3\)](#).

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 [EMR_SELECTOBJECT](#) record (section 2.3.8.5).

000034A0:	25 00 00 00 0C 00 00 00
000034B0:03	00 00 00 00

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000025)																																
Size (0x0000000C)																																
ihObject (0x00000003)																																

Figure 27: EMR_SELECTOBJECT Record Example

Type: Identifies this record type as EMR_SELECTOBJECT.

Size: Specifies the size of this record, in bytes.

ihObject: Specifies the index of an object in the [EMF Object Table](#).

3.2.13 EMR_EXTCREATEFONTINDIRECTW Example

This section provides an example of the [EMR_EXTCREATEFONTINDIRECTW](#) record (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 08 00 00 3B 09 00 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 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 01 00 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 20 AC CF 02 64 76 00 08
00003620:00 00 00 00

```

0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000004)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

Figure 28: 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 that is assigned to the font.

elw: To determine the type of logical font object in this field, an algorithm (section [2.3.7.8](#)) is applied, which indicates this is a variable-length [LogFontExDv object \(section 2.2.15\)](#).

0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	3	0	1
Height (0xFFFFFFF2)																															
Width (0x00000000)																															
Escapement (0x000000C4E)																															
Orientation (0x000000C4E)																															
Weight (0x0000000C8)																															
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 29: LogFontExDv object, part 1

Height: 0xFFFFFFF2 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: 0x000000C4E 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_CHARSET**, as defined in the WMF [CharacterSet](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.5).

OutPrecision: 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the WMF [OutPrecision](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.21). The value 0x04 specifies 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](#) flags ([\[MS-WMF\]](#) section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality: 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration ([\[MS-WMF\]](#) section 2.1.1.10).

PitchAndFamily: 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [PitchAndFamily object](#) ([\[MS-WMF\]](#) section 2.2.2.14).

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 30: 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: 0x08007664 specifies the signature of a [DesignVector object \(section 2.2.3\)](#).

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 [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)).

00003620: 25 00 00 00 0C 00 00 00 04 00 00 00 00

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
Type (0x00000025)																																		
Size (0x0000000C)																																		
ihObject (0x00000004)																																		

Figure 31: EMR_SELECTOBJECT record example

Type: Identifies this record type as EMR_SELECTOBJECT.

Size: Specifies the size of this record, in bytes.

ihObject: Specifies the index of an object in the [EMF Object Table](#).

3.2.15 EMR_DELETEOBJECT Example

This section provides an example of the [EMR_DELETEOBJECT](#) record (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	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
Type (0x00000028)																																		
Size (0x0000000C)																																		
ihObject (0x00000003)																																		

Figure 32: EMR_DELETEOBJECT record example

Type: Identifies this record type as EMR_DELETEOBJECT.

Size: Specifies the size of this record, in bytes.

ihObject: Specifies the index of the object to be deleted.

3.2.16 EMR_EXTCREATEFONTINDIRECTW Example

This section provides an example of the [EMR_EXTCREATEFONTINDIRECTW](#) record (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 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 00  
00003690:00 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 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 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 00  
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00 00  
00003720:65 58 53 FE 00 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 00  
00003750:00 00 00 00 FE 07 00 00 79 0D 21 11 00 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 F5 13 00 00 00 00 00 00 00  
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00 00  
00003790:04 00 00 00 00 00 00 00 24 07 5A FE FE 07 00 00 00  
000037A0:01 00 00 00 64 76 00 08 00 00 00 00 00 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000052)																				
Size (0x000000170)																				
ihFonts (0x000000003)																				
elw (variable)																				
...																				
(elw cont'd for 87 rows)																				

Figure 33: EMR_EXTCREATEFONTINDIRECTW record example

Type: 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size: 0x000000170 specifies the size of this record, in bytes.

ihFonts: 0x000000003 specifies the object index that is assigned to the font.

elw: To determine the type of logical font object in this field, an algorithm (section [2.3.7.8](#)) is applied, which indicates this is a variable-length [LogFontExDv object \(section 2.2.15\)](#).

0	1	2	3	4	5	6	7	8	9	1	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 ("Microsoft Sans Serif")																																
...																																
(Facename cont'd for 14 rows)																																

Figure 34: 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_CHARSET**, as defined in the WMF **CharacterSet** enumeration ([\[MS-WMF\]](#) section 2.1.1.5).

OutPrecision: 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the WMF **OutPrecision** enumeration ([\[MS-WMF\]](#) section 2.1.1.21). The value 0x04 specifies 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** flags ([\[MS-WMF\]](#) section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality: 0x00 specifies default output quality, from the WMF **FontQuality** enumeration ([\[MS-WMF\]](#) section 2.1.1.10).

PitchAndFamily: 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [PitchAndFamily object](#) ([\[MS-WMF\]](#) section 2.2.2.14).

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	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 35: 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 a [DesignVector object](#) (section 2.2.3).

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 [EMR_SELECTOBJECT](#) record (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	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000025)																														
Size (0x0000000C)																														
ihObject (0x00000003)																														

Figure 36: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_SELECTOBJECT](#) record (section [2.3.8.5](#)).

000037B0:	25 00 00 00 0C 00 00 00 00
000037C0:02	00 00 00 00

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
Type (0x00000025)																																		
Size (0x0000000C)																																		
ihObject (0x00000002)																																		

Figure 37: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_DELETEOBJECT](#) record (section [2.3.8.3](#)).

000037C0:	28 00 00 00 0C 00 00 00 00 04 00 00 00 00
-----------	---

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
Type (0x00000028)																																		
Size (0x0000000C)																																		
ihObject (0x00000004)																																		

Figure 38: EMR_DELETEOBJECT record example

Type: 0x00000028 identifies this record type as EMR_DELETEOBJECT.

Size: 0x0000000C specifies the size of this record, in bytes.

ihObject: 0x00000004 specifies the index of the object to be deleted.

3.2.20 EMR_DELETEOBJECT Example

This section provides an example of the [EMR_DELETEOBJECT](#) record (section [2.3.8.3](#)).

000037D0:28 00 00 00 0C 00 00 00 03 00 00 00 00

0	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	2	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000028)																														
Size (0x0000000C)																														
ihObject (0x00000003)																														

Figure 39: EMR_DELETEOBJECT record example

Type: 0x00000028 identifies this record type as EMR_DELETEOBJECT.

Size: 0x0000000C specifies the size of this record, in bytes.

ihObject: 0x00000003 specifies the index of the object to be deleted.

3.2.21 EMR_SELECTOBJECT Example

This section provides an example of the [EMR_SELECTOBJECT](#) record (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	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x8000000D = SYSTEM_FONT)																															

Figure 40: EMR_SELECTOBJECT record example

Type: 0x00000025 identifies this record type as EMR_SELECTOBJECT.

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 [EMR_EOF](#) record (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	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
Type (0x0000000E)																																		
Size (0x00000014)																																		
nPalEntries (0x00000000)																																		
offPalEntries (0x00000010)																																		
SizeLast (0x00000014)																																		

Figure 41: EMR_EOF record example

Type: 0x0000000E identifies this record type as EMR_EOF.

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 the **Size** value. It is the last field in the metafile.

4 Security Considerations

This file format enables third parties to send payloads (such as PostScript) to pass through as executable code.

5 Appendix A: Product Behavior

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

- Windows NT 3.1 operating system
- Windows NT 3.5 operating system
- Windows NT 3.51 operating system
- Windows NT 4.0 operating system
- Microsoft Windows 98 operating system
- Windows Millennium Edition operating system
- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

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

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

[<1> Section 1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition do not support WCS color management.

[<2> Section 1.3.1:](#) EMF metafiles have changed over the years with the evolution of Windows operating systems. They were introduced with 32-bit Windows and replaced the 16-bit WMF metafile [\[MS-WMF\]](#) as the Windows standard.

[<3> Section 1.3.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, and Windows Millennium Edition: This form of EMF metafile is not supported.

[<4> Section 1.3.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, Windows Millennium Edition, Windows 2000: This form of EMF metafile is not supported.

[<5> Section 1.4:](#) The EMF format supersedes WMF format, which was used in 16-bit Windows versions.

[<6> Section 1.4:](#) Windows applications that use the **Graphics Device Interface, Extended (GDI) +** API can create EMF metafiles that contain EMF+ records. See [\[MSDN-GDI+\]](#) for more information.

[<7> Section 1.6:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: EMF extension 1 is not supported.

[<8> Section 1.6:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, Windows Millennium Edition, Windows NT 4.0, and Windows 2000: EMF extension 2 is not supported.

[<9> Section 2.1.1:](#) Only Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 support **EMR_SETMAPPERFLAGS**.

[<10> Section 2.1.1:](#) Windows **GDI** emulates [EMR_EXTTEXTOUTA](#) with an [EMR_EXTTEXTOUTW](#) record.

[<11> Section 2.1.1:](#) Windows NT 3.1 is the only Windows version in which GDI uses [EMR_POLYTEXTOUTA](#) records for text output. All other versions emulate [EMR_POLYTEXTOUTA](#) with [EMR_EXTTEXTOUTW](#) records.

[<12> Section 2.1.1:](#) Windows NT 3.1 is the only Windows version in which GDI uses [EMR_POLYTEXTOUTW](#) records for text output. All other versions emulate [EMR_POLYTEXTOUTW](#) with [EMR_EXTTEXTOUTW](#) records.

[<13> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_SETICMMODE**.

[<14> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_CREATECOLORSPACE**.

[<15> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_SETCOLORSPACE**.

[<16> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_DELETECOLORSPACE**.

[<17> Section 2.1.1:](#) Windows uses an [EMR_DELETEOBJECT](#) record to delete a logical color space object.

[<18> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_GLSRECORD**.

[<19> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_GLSBOUNDEDRECORD**.

[<20> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_PIXELFORMAT**.

[<21> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_COLORCORRECTPALETTE**.

[<22> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETICMPROFILEA**.

[<23> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETICMPROFILEW**.

[<24> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_ALPHABLEND**.

[<25> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETLAYOUT**.

[<26> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_TRANSPARENTBLT**.

[<27> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_GRADIENTFILL**.

[<28> Section 2.1.1:](#) Windows GDI uses an **EMR_EXTEXTOUTW** record (section [2.3.5.8](#)) to perform this function.

[<29> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_COLORMATCHTOTARGETW**.

[<30> Section 2.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_CREATECOLORSPACEW**.

[<31> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support the **ETO_GLYPH_INDEX** flag used for bitmap and vector fonts—in addition to TrueType fonts—to indicate that no further language processing is necessary and that GDI should process the string directly. See [MSDN-GDI+](#) for more information.

[<32> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support the **ETO_RTLREADING** flag used to indicate right-to-left reading order.

[<33> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support the **ETO_NUMERICSLATIN** flag used to indicate the display of numeric digits appropriate to the locale.

[<34> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support the **ETO_NUMERICSLATIN** flag used to indicate the display of numeric digits appropriate to Europe.

[<35> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support the **ETO_IGNORELANGUAGE** flag used to indicate that international scripting support is not used, which may cause no text to be output.

[<36> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, Windows Millennium Edition, Windows NT 4.0, and Windows 2000: Do not support the **ETO_PDY** flag used to indicate that both horizontal and vertical character displacement values SHOULD be provided.

[<37> Section 2.1.11:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, Windows Millennium Edition, Windows NT 4.0, and Windows 2000: Do not support the **ETO_REVERSE_INDEX_MAP** flag.

[<38> Section 2.1.16:](#) **GM_COMPATIBLE** graphics mode is used for compatibility between both 16-bit and 32-bit systems.

[<39> Section 2.1.16:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, Windows Millennium Edition, Windows NT 4.0, and Windows 2000: **GM_ADVANCED** is not supported.

[<40> Section 2.1.18:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support **ICM_OFF**.

[<41> Section 2.1.18:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support **ICM_ON**.

[<42> Section 2.1.18:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support **ICM_QUERY**.

[<43> Section 2.1.18:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: Do not support **ICM_DONE_OUTSIDEDC**.

[<44> Section 2.1.31:](#) In Windows implementations, **BS_HOLLOW** was added as a duplicate symbolic name for **BS_NULL** because **BS_NULL** was too easily mistaken for a null pointer.

BS_HOLLOW is used by an application when GDI requires a non-null brush parameter, but the application requires that no brush be used. See [\[MSDN-GDI+\]](#) for more information.

[<45> Section 2.1.31:](#) On Windows, this is the "Courier" font.

[<46> Section 2.1.31:](#) On Windows, this is the "MS Sans Serif" font.

[<47> Section 2.1.31:](#) On Windows, this is the font used to draw menu text and dialog box controls.

Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, and Windows Millennium Edition: The system font is "MS Sans Serif".

Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2: The system font is "Tahoma".

[<48> Section 2.1.31:](#) Windows 98 and Windows Millennium Edition: This value is not supported.

Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2: This value is considered equivalent to **SYSTEM_FONT** for the purposes of screen display of metafiles.

[<49> Section 2.1.31:](#) On Windows, this palette consists of the static colors in the system palette.

[<50> Section 2.1.31:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, and Windows Millennium Edition: The default user interface font is "MS Sans Serif".

Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2: The default user-interface font is "Tahoma".

[<51> Section 2.1.31:](#) Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2: The default brush is **WHITE_BRUSH**.

[<52> Section 2.1.31:](#) Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2: The default pen is **BLACK_PEN**.

[<53> Section 2.1.32:](#) Windows also uses the following symbolic names for the [StretchMode](#) enumeration; their meanings are exactly the same as the members with the same values.

```
#define BLACKONWHITE          1
#define WHITEONBLACK          2
#define COLORONCOLOR          3
#define HALFTONE               4
```

[<54> Section 2.2.2:](#) Windows 98 and Windows Millennium Edition do not support the [ColorAdjustment](#) object.

[<55> Section 2.2.2:](#) Windows can generate [ColorAdjustment](#) objects with values outside their valid ranges. Such objects are ignored.

[<56> Section 2.2.5:](#) In Windows implementations, this is the clipping and/or opaquing rectangle that is passed to GDI methods **ExtTextOutA** and **ExtTextOutW**.

[<57> Section 2.2.6:](#) Windows does not parse the PostScript data in an [EpsData object](#); the data is handed off to the graphics printer driver if the driver supports PostScript printing.

[<58> Section 2.2.9:](#) Windows does not check this value.

[<59> Section 2.2.12:](#) In Windows implementations, **BS_HOLLOW** was added as a duplicate symbolic name for **BS_NULL** because **BS_NULL** was too easily mistaken for a null pointer.

BS_HOLLOW is used by an application when GDI requires a non-null brush parameter, but the application requires that no brush be used. See [\[MSDN-GDI+\]](#) for more information.

[<60> Section 2.2.13:](#) 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.

[<61> Section 2.2.13:](#) Windows uses a weight value of 400 by default.

Value	Weight
Thin	100
Extra Light (Ultra Light)	200
Light	300
Normal (Regular)	400
Medium	500
Semi-Bold (Demi-Bold)	600
Bold	700
Extra Bold (Ultra Bold)	800

Value	Weight
Heavy (Black)	900

[<62> Section 2.2.20:](#) In Windows implementations, **BS_HOLLOW** was added as a duplicate symbolic name for **BS_NULL** because **BS_NULL** was too easily mistaken for a null pointer.

BS_HOLLOW is used by an application when GDI requires a non-null brush parameter, but the application requires that no brush be used. See [\[MSDN-GDI+\]](#) for more information.

[<63> Section 2.2.22:](#) Windows implementations do not support this flag.

[<64> Section 2.2.22:](#) Windows can use this flag to indicate that the pixel format specified by this structure is supported by GDI. See [\[MSDN-GDI+\]](#) for more information.

Windows can also use this flag to specify single-buffering for the pixel buffer.

[<65> Section 2.2.22:](#) Windows uses this flag to indicate that the pixel pixel format is supported by GDI.

[<66> Section 2.2.22:](#) Windows uses this with OpenGL drawing only.

Windows NT 3.1, Windows NT 3.51, Windows NT 4.0, Windows 98, Windows 2000, Windows Millennium Edition, Windows XP, and Windows Server 2003 do not support this flag.

[<67> Section 2.2.22:](#) Windows does not support alpha bitplanes.

[<68> Section 2.2.22:](#) Windows does not support alpha bitplanes.

[<69> Section 2.2.22:](#) Windows does not support alpha bitplanes.

[<70> Section 2.2.27:](#) In this case, Windows uses the logical font that is currently selected in the playback device context.

[<71> Section 2.3.1:](#) Windows does not support rotation or shear transforms.

[<72> Section 2.3.1.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_ALPHABLEND**.

[<73> Section 2.3.1.3:](#) Windows might set this to a non-zero value.

[<74> Section 2.3.1.8:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_TRANSPARENTBLT**.

[<75> Section 2.3.1.8:](#) Windows uses the [EMR_ALPHABLEND](#) record (section [2.3.1.1](#)) to specify a block transfer of a 32 bits-per-pixel bitmap with alpha transparency.

[<76> Section 2.3.3.1:](#) Windows NT 3.1, Windows NT 3.51, and Windows NT 4.0 ignore [EMR_COMMENT](#) records.

[<77> Section 2.3.3.4.3:](#) On playback, the first graphics format recognized by Windows is used to render the image.

[<78> Section 2.3.4.2:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition create EMF metafiles with an [EmfMetafileHeader](#) header record (section [2.3.4.2.1](#)).

[<79> Section 2.3.4.2:](#) Windows NT 4.0 creates EMF metafiles with an [EmfMetafileHeaderExtension1](#) header record (section [2.3.4.2.2](#)).

[<80> Section 2.3.4.2:](#) Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 create EMF metafiles with an [EmfMetafileHeaderExtension2](#) header record (section [2.3.4.2.3](#)).

[<81> Section 2.3.4.2.2:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition do not support this type of EMF metafile header.

[<82> Section 2.3.4.2.3:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, and Windows Millennium Edition do not support this type of EMF metafile header.

[<83> Section 2.3.5.7:](#) Windows GDI emulates [EMR_EXTTEXTOUTA](#) with an [EMR_EXTTEXTOUTW](#) record.

[<84> Section 2.3.5.12:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support [EMR_GRADIENTFILL](#).

[<85> Section 2.3.5.12:](#) Windows uses true colors in 24-bits-per-pixel (bpp) and 32-bpp formats, and [dithering](#) in 4-bpp, 8-bpp, and 16-bpp formats.

[<86> Section 2.3.5.32:](#) Windows NT 3.1 is the only Windows version in which GDI uses [EMR_POLYTEXTOUTA](#) records for text output. All other versions emulate [EMR_POLYTEXTOUTA](#) with [EMR_EXTTEXTOUTW](#) records.

[<87> Section 2.3.5.33:](#) Windows NT 3.1 is the only Windows version in which GDI uses [EMR_POLYTEXTOUTW](#) records for text output. All other versions emulate [EMR_POLYTEXTOUTW](#) with [EMR_EXTTEXTOUTW](#) records.

[<88> Section 2.3.7.2:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support [EMR_CREATECOLORSPACE](#).

[<89> Section 2.3.7.3:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support [EMR_CREATECOLORSPACEW](#).

[<90> Section 2.3.7.8:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 [metafiles](#) contain a [LogFont](#) object in this field.

[<91> Section 2.3.8.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support [EMR_COLORCORRECTPALETTE](#).

[<92> Section 2.3.8.2:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support [EMR_DELETECOLORSPACE](#).

[<93> Section 2.3.8.2:](#) Windows uses an [EMR_DELETEOBJECT](#) record to delete a logical color space object.

[<94> Section 2.3.8.4:](#) Windows GDI does not perform parameter validation on this value, which can lead to the generation of EMF metafiles that contain invalid [EMR_RESIZEPALETTE](#) records. Windows ignores such invalid records when processing metafiles.

[<95> Section 2.3.8.7:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support [EMR_SETCOLORSPACE](#).

[<96> Section 2.3.9:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: OpenGL records are not supported.

[<97> Section 2.3.9.1:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_GLSBOUNDEDRECORD**.

[<98> Section 2.3.9.2:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_GLSRECORD**.

[<99> Section 2.3.11.1:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_COLORMATCHTOTARGETW**.

[<100> Section 2.3.11.1:](#) On Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2, before applying the current color transform, WCS is enabled in the playback device context.

[<101> Section 2.3.11.5:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_PIXELFORMAT**.

[<102> Section 2.3.11.14:](#) Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_SETICMMODE**.

[<103> Section 2.3.11.15:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETICMPROFILEA**.

[<104> Section 2.3.11.16:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETICMPROFILEW**.

[<105> Section 2.3.11.17:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not support **EMR_SETLAYOUT**.

[<106> Section 2.3.11.20:](#) Only Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 support **EMR_SETMAPPERFLAGS**.

[<107> Section 2.3.11.21:](#) Windows GDI accepts a floating-point value for the corresponding miter length limit value,

[<108> Section 2.3.11.27:](#) Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, Windows 98, and Windows Millennium Edition do not support **EMR_SETTEXTJUSTIFICATION**.

[<109> Section 2.3.11.27:](#) Windows GDI uses an [EMR_EXTTEXTOUTW](#) record (section [2.3.5.8](#)) to perform this function.

6 Change Tracking

This section identifies changes that were made to the [MS-EMF] protocol document between the February 2014 and May 2014 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.

- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.3.5.12 EMR_GRADIENTFILL Record	70856 Added the VertexPadding field to the VertexData structure.	Y	Content updated.

7 Index

A

[Applicability](#) 20
[ArcDirection enumeration](#) 30
[ArmStyle enumeration](#) 31

B

[BackgroundMode enumeration](#) 32
[BitFIX28_4 Object](#) 53
[BitFIX28_4 packet](#) 53
[BitmapRecordTypes packet](#) 83
[Byte ordering](#) 20
[Byte ordering example](#) 236

C

[Change tracking](#) 302
[ClippingRecordTypes packet](#) 114
[ColorAdjustment enumeration](#) 32
[ColorAdjustment Object](#) 53
[ColorAdjustment packet](#) 53
[ColorMatchToTarget enumeration](#) 32
[ColorSpace enumeration](#) 33
[CommentRecordTypes packet](#) 118
[Contrast enumeration](#) 33
[ControlRecordTypes packet](#) 127

D

[DesignVector Object](#) 55
[DesignVector packet](#) 55
[DIBColors enumeration](#) 34
[Drawing Record Types](#) 139
[DrawingRecordTypes packet](#) 139

E

[EMF metafile example](#) 237
[EmfMetafileHeader packet](#) 133
[EmfMetafileHeader Record](#) 133
[EmfMetafileHeaderExtension1 packet](#) 134
[EmfMetafileHeaderExtension1 Record](#) 134
[EmfMetafileHeaderExtension2 packet](#) 136
[EmfMetafileHeaderExtension2 Record](#) 136
[EMR_ALPHABLEND packet](#) 86
[EMR_ALPHABLEND record](#) 86
[EMR_ANGLEARC packet](#) 143
[EMR_ANGLEARC Record](#) 143
[EMR_ARC packet](#) 144
[EMR_ARC Record](#) 144
[EMR_ARCTO packet](#) 145
[EMR_ARCTO Record](#) 145
[EMR_BITBLT example](#) ([section 3.2.4](#) 256, [section 3.2.6](#) 259)
[EMR_BITBLT packet](#) 90
[EMR_BITBLT Record](#) 90
[EMR_CHORD packet](#) 146

[EMR_CHORD Record](#) 146
[EMR_COLORCORRECTPALETTE packet](#) 198
[EMR_COLORCORRECTPALETTE record](#) 198
[EMR_COLORMATCHTOTARGETW packet](#) 211
[EMR_COLORMATCHTOTARGETW record](#) 211
[EMR_COMMENT packet](#) 120
[EMR_COMMENT Record](#) 120
[EMR_COMMENT_BEGINGROUP packet](#) 123
[EMR_COMMENT_BEGINGROUP Record](#) 123
[EMR_COMMENT_EMFPLUS packet](#) 120
[EMR_COMMENT_EMFPLUS Record](#) 120
[EMR_COMMENT_EMFSPOOL packet](#) 121
[EMR_COMMENT_EMFSPOOL Record](#) 121
[EMR_COMMENT_ENDGROUP packet](#) 124
[EMR_COMMENT_ENDGROUP Record](#) 124
[EMR_COMMENT_MULTIFORMATS packet](#) 125
[EMR_COMMENT_MULTIFORMATS Record](#) 125
[EMR_COMMENT_PUBLIC packet](#) 122
[EMR_COMMENT_PUBLIC Record Types](#) 122
[EMR_COMMENT_WINDOWS_METAFILE packet](#) 126
[EMR_COMMENT_WINDOWS_METAFILE Record](#) 126
[EMR_CREATEBRUSHINDIRECT example](#) 255
[EMR_CREATEBRUSHINDIRECT packet](#) 186
[EMR_CREATEBRUSHINDIRECT Record](#) 186
[EMR_CREATECOLORSPACE packet](#) 187
[EMR_CREATECOLORSPACE record](#) 187
[EMR_CREATECOLORSPACEW packet](#) 188
[EMR_CREATECOLORSPACEW record](#) 188
[EMR_CREATEDIBPATTERNBRUSHPT packet](#) 189
[EMR_CREATEDIBPATTERNBRUSHPT Record](#) 189
[EMR_CREATEMONOBRAHS packet](#) 190
[EMR_CREATEMONOBRAHS Record](#) 190
[EMR_CREATEPALETTE packet](#) 192
[EMR_CREATEPALETTE Record](#) 192
[EMR_CREATEPEN packet](#) 193
[EMR_CREATEPEN Record](#) 193
[EMR_DELETECOLORSPACE packet](#) 199
[EMR_DELETECOLORSPACE record](#) 199
[EMR_DELETEOBJECT example](#) ([section 3.2.15](#) 286, [section 3.2.19](#) 290, [section 3.2.20](#) 291)
[EMR_DELETEOBJECT packet](#) 200
[EMR_DELETEOBJECT Record](#) 200
[EMR_DRAWESCAPE packet](#) 182
[EMR_DRAWESCAPE Record](#) 182
[EMR_ELLIPSE packet](#) 147
[EMR_ELLIPSE Record](#) 147
[EMR_EOF example](#) 291
[EMR_EOF packet](#) 128
[EMR_EOF Record](#) 128
[EMR_EXCLUDECLIPRECT packet](#) 115
[EMR_EXCLUDECLIPRECT Record](#) 115
[EMR_EXTCREATEFONTINDIRECTW example](#) ([section 3.2.8](#) 274, [section 3.2.11](#) 279, [section 3.2.13](#) 283, [section 3.2.16](#) 287)
[EMR_EXTCREATEFONTINDIRECTW packet](#) 193
[EMR_EXTCREATEFONTINDIRECTW Record](#) 193
[EMR_EXTCREATEPEN packet](#) 195
[EMR_EXTCREATEPEN Record](#) 195

EMR_EXTEscape packet	183
EMR_EXTEscape Record	183
EMR_EXTFLOODFILL packet	147
EMR_EXTFLOODFILL Record	147
EMR_EXTSELECTCLIPRGN packet	116
EMR_EXTSELECTCLIPRGN Record	116
EMR_EXTTEXTOUTA packet	148
EMR_EXTTEXTOUTA record	148
EMR_EXTTEXTOUTW example	278
EMR_EXTTEXTOUTW packet	149
EMR_EXTTEXTOUTW Record	149
EMR_FILLPATH packet	150
EMR_FILLPATH Record	150
EMR_FILLRGN packet	151
EMR_FILLRGN Record	151
EMR_FORCEUFIMAPPING packet	212
EMR_FORCEUFIMAPPING Record	212
EMR_FRAMERGN packet	152
EMR_FRAMERGN Record	152
EMR_GLSBOUNDEDRECORD packet	205
EMR_GLSBOUNDEDRECORD record	205
EMR_GLSRECORD packet	205
EMR_GLSRECORD record	205
EMR_GRADIENTFILL packet	153
EMR_GRADIENTFILL record	153
EMR_HEADER example	252
EMR_HEADER packet	129
EMR_HEADER Record Types	129
EMR_INTERSECTCLIPRECT packet	117
EMR_INTERSECTCLIPRECT Record	117
EMR_INVERTRGN packet	213
EMR_INVERTRGN Record	213
EMR_LINETO packet	155
EMR_LINETO Record	155
EMR_MASKBLT packet	93
EMR_MASKBLT Record	93
EMR_MODIFYWORLDTRANSFORM packet	232
EMR_MODIFYWORLDTRANSFORM Record	232
EMR_MOVETOEX packet	214
EMR_MOVETOEX Record	214
EMR_NAMEDESCAPE packet	183
EMR_NAMEDESCAPE Record	183
EMR_OFFSETCLIPRGN packet	117
EMR_OFFSETCLIPRGN Record	117
EMR_PAINTRGN packet	155
EMR_PAINTRGN Record	155
EMR_PIE packet	156
EMR_PIE Record	156
EMR_PIXELFORMAT packet	214
EMR_PIXELFORMAT record	214
EMR_PLGBLT packet	98
EMR_PLGBLT Record	98
EMR_POLYBEZIER packet	157
EMR_POLYBEZIER Record	157
EMR_POLYBEZIER16 packet	158
EMR_POLYBEZIER16 Record	158
EMR_POLYBEZIERTO packet	159
EMR_POLYBEZIERTO Record	159
EMR_POLYBEZIERTO16 packet	160
EMR_POLYBEZIERTO16 Record	160
EMR_POLYDRAW packet	161
EMR_POLYDRAW Record	161
EMR_POLYDRAW16 packet	162
EMR_POLYDRAW16 Record	162
EMR_POLYGON packet	163
EMR_POLYGON Record	163
EMR_POLYGON16 packet	164
EMR_POLYGON16 Record	164
EMR_POLYLINE packet	165
EMR_POLYLINE Record	165
EMR_POLYLINE16 packet	166
EMR_POLYLINE16 Record	166
EMR_POLYLINETO packet	167
EMR_POLYLINETO Record	167
EMR_POLYPOLYGON packet	169
EMR_POLYPOLYGON Record	169
EMR_POLYPOLYGON16 packet	170
EMR_POLYPOLYGON16 Record	170
EMR_POLYPOLYLINE packet	171
EMR_POLYPOLYLINE Record	171
EMR_POLYPOLYLINE16 packet	172
EMR_POLYPOLYLINE16 Record	172
EMR_POLYTEXTOUTA packet	174
EMR_POLYTEXTOUTA record	174
EMR_POLYTEXTOUTW packet	175
EMR_POLYTEXTOUTW record	175
EMR_RECTANGLE packet	176
EMR_RECTANGLE Record	176
EMR_RESIZEPALETTE packet	200
EMR_RESIZEPALETTE Record	200
EMR_RESTOREDC packet	215
EMR_RESTOREDC Record	215
EMR_ROUNDRECT packet	177
EMR_ROUNDRECT Record	177
EMR_SCALEVIEWPORTEXTEX packet	216
EMR_SCALEVIEWPORTEXTEX Record	216
EMR_SCALEWINDOWEXTEX packet	216
EMR_SCALEWINDOWEXTEX Record	216
EMR_SELECTCLIPPATH packet	118
EMR_SELECTCLIPPATH Record	118
EMR_SELECTOBJECT example	(section 3.2.3 256, section 3.2.5 258, section 3.2.9 277, section 3.2.12 282, section 3.2.14 286, section 3.2.17 289, section 3.2.18 290, section 3.2.21 291)
EMR_SELECTOBJECT packet	201
EMR_SELECTOBJECT Record	201
EMR_SELECTPALETTE packet	201
EMR_SELECTPALETTE Record	201
EMR_SETARCDIRECTION packet	217
EMR_SETARCDIRECTION Record	217
EMR_SETBKCOLOR packet	218
EMR_SETBKCOLOR Record	218
EMR_SETBKMODE example	274
EMR_SETBKMODE packet	219
EMR_SETBKMODE Record	219
EMR_SETBRUSHORGEX packet	219
EMR_SETBRUSHORGEX Record	219
EMR_SETCOLORADJUSTMENT packet	220
EMR_SETCOLORADJUSTMENT Record	220
EMR_SETCOLORSPACE packet	202

[EMR_SETCOLORSPACE record](#) 202
[EMR_SETDBITSTODEVICE packet](#) 102
[EMR_SETDBITSTODEVICE Record](#) 102
[EMR_SETICMMODE packet](#) 220
[EMR_SETICMMODE record](#) 220
[EMR_SETICMPROFILEA packet](#) 221
[EMR_SETICMPROFILEA record](#) 221
[EMR_SETICMPROFILEW packet](#) 222
[EMR_SETICMPROFILEW record](#) 222
[EMR_SETLAYOUT packet](#) 223
[EMR_SETLAYOUT record](#) 223
[EMR_SETLINKEDUFIS packet](#) 223
[EMR_SETLINKEDUFIS Record](#) 223
[EMR_SETMAPMODE packet](#) 224
[EMR_SETMAPMODE Record](#) 224
[EMR_SETMAPPERFLAGS packet](#) 225
[EMR_SETMAPPERFLAGS record](#) 225
[EMR_SETMITERLIMIT packet](#) 225
[EMR_SETMITERLIMIT Record](#) 225
[EMR_SETPALETTEENTRIES packet](#) 203
[EMR_SETPALETTEENTRIES Record](#) 203
[EMR_SETPIXELV packet](#) 177
[EMR_SETPIXELV Record](#) 177
[EMR_SETPOLYFILLMODE packet](#) 226
[EMR_SETPOLYFILLMODE Record](#) 226
[EMR_SETROP2 packet](#) 226
[EMR_SETROP2 Record](#) 226
[EMR_SETSTRETCHBLTMODE packet](#) 227
[EMR_SETSTRETCHBLTMODE Record](#) 227
[EMR_SETTEXTALIGN packet](#) 228
[EMR_SETTEXTALIGN Record](#) 228
[EMR_SETTEXTCOLOR record](#) 228
[EMR_SETTEXTJUSTIFICATION packet](#) 229
[EMR_SETVIEWPORTEXTEX packet](#) 229
[EMR_SETVIEWPORTEXTEX Record](#) 229
[EMR_SETVIEWPORTORGEX packet](#) 230
[EMR_SETVIEWPORTORGEX Record](#) 230
[EMR_SETWINDOWEXTEX packet](#) 230
[EMR_SETWINDOWEXTEX Record](#) 230
[EMR_SETWINDOWORGEX packet](#) 231
[EMR_SETWINDOWORGEX Record](#) 231
[EMR_SETWORLDTRANSFORM packet](#) 233
[EMR_SETWORLDTRANSFORM Record](#) 233
[EMR_SMALLTEXTOUT packet](#) 178
[EMR_SMALLTEXTOUT Record](#) 178
[EMR_STRETCHBLT packet](#) 104
[EMR_STRETCHBLT Record](#) 104
[EMR_STRETCHDIBITS packet](#) 108
[EMR_STRETCHDIBITS Record](#) 108
[EMR_STROKEANDFILLPATH packet](#) 180
[EMR_STROKEANDFILLPATH Record](#) 180
[EMR_STROKEPATH packet](#) 180
[EMR_STROKEPATH Record](#) 180
[EMR_TRANSPARENTBLT packet](#) 111
[EMR_TRANSPARENTBLT record](#) 111
[EmrComment enumeration](#) 34
[EmrFormat Object](#) 55
[EmrFormat packet](#) 55
[EmrText Object](#) 56
[EmrText packet](#) 56
[Enumerations](#) 22

[EpsData Object](#) 58
[EpsData packet](#) 58
[EscapeRecordTypes packet](#) 181
[Examples](#)
[byte ordering example](#) 236
[EMF metafile example](#) 237
[EMR_BITBLT example \(section 3.2.4\)](#) 256, [section 3.2.6](#) 259
[EMR_CREATEBRUSHINDIRECT example](#) 255
[EMR_DELETEOBJECT example \(section 3.2.15\)](#)
286, [section 3.2.19](#) 290, [section 3.2.20](#) 291
[EMR_EOF example](#) 291
[EMR_EXTCREATEFONTINDIRECTW example \(section 3.2.8\)](#) 274, [section 3.2.11](#) 279, [section 3.2.13](#) 283, [section 3.2.16](#) 287
[EMR_EXTEXTOUTW example](#) 278
[EMR_HEADER example](#) 252
[EMR_SELECTOBJECT example \(section 3.2.3\)](#) 256, [section 3.2.5](#) 258, [section 3.2.9](#) 277, [section 3.2.12](#) 282, [section 3.2.14](#) 286, [section 3.2.17](#) 289, [section 3.2.18](#) 290, [section 3.2.21](#) 291
[EMR_SETBKMODE example](#) 274
[metafile design examples](#) 235
[ExtTextOutOptions enumeration](#) 35

F

[FamilyType enumeration](#) 36
[Fields - vendor-extensible](#) 21
[FloodFill enumeration](#) 36
[FormatSignature enumeration](#) 36

G

[Glossary](#) 10
[GradientFill enumeration](#) 37
[GradientRectangle Object](#) 59
[GradientRectangle packet](#) 59
[GradientTriangle Object](#) 59
[GradientTriangle packet](#) 59
[GraphicsMode enumeration](#) 37

H

[HatchStyle enumeration](#) 38
[Header Object](#) 60
[Header packet](#) 60
[HeaderExtension1 Object](#) 62
[HeaderExtension1 packet](#) 62
[HeaderExtension2 Object](#) 62
[HeaderExtension2 packet](#) 62

I

[ICMMode enumeration](#) 39
[Illuminant enumeration](#) 39
[Informative references](#) 17
[Introduction](#) 10

L

[Letterform enumeration](#) 40

[Localization](#) 20
[LogBrushEx Object](#) 63
[LogBrushEx packet](#) 63
[LogFont Object](#) 63
[LogFont packet](#) 63
[LogFontEx Object](#) 66
[LogFontEx packet](#) 66
[LogFontExDv Object](#) 68
[LogFontExDv packet](#) 68
[LogFontPanose Object](#) 69
[LogFontPanose packet](#) 69
[LogPalette Object](#) 71
[LogPalette packet](#) 71
[LogPaletteEntry Object](#) 71
[LogPaletteEntry packet](#) 71
[LogPen Object](#) 72
[LogPen packet](#) 72
[LogPenEx Object](#) 72
[LogPenEx packet](#) 72

M

[MapMode enumeration](#) 41
[Metafile design examples](#) 235
[Metafile structure](#) 17
[MetafileVersion enumeration](#) 42
[MidLine enumeration](#) 42
[ModifyWorldTransformMode enumeration](#) 43
[MR_SETTEXTCOLOR packet](#) 228

N

[Normative references](#) 17

O

[ObjectCreationRecordTypes packet](#) 184
[ObjectManipulationRecordTypes packet](#) 197
[Objects](#) 53
[OpenGLRecordTypes packet](#) 203
[Overview \(synopsis\)](#) 17

P

[Panose Object](#) 74
[Panose packet](#) 74
[Path Bracket Record Types](#) 206
[PathBracketRecordTypes packet](#) 206
[PenStyle enumeration](#) 43
[PixelFormatDescriptor Object](#) 75
[PixelFormatDescriptor packet](#) 75
[Point enumeration](#) 45
[Point28_4 Object](#) 78
[Point28_4 packet](#) 78
[PolygonFillMode enumeration](#) 45
[Product behavior](#) 294
[Proportion enumeration](#) 45

R

[RecordType enumeration](#) 22
References

[informative](#) 17
[normative](#) 17
[RegionData Object](#) 79
[RegionData packet](#) 79
[RegionDataHeader Object](#) 79
[RegionDataHeader packet](#) 79
[RegionMode enumeration](#) 46
[Relationship to other protocols](#) 20

S

[Security](#) 293
[SerifType enumeration](#) 47
[StateRecordTypes packet](#) 207
[StockObject enumeration](#) 48
[StretchMode enumeration](#) 50
[StrokeVariation enumeration](#) 51
Structures
[Drawing Record Types](#) 139
[enumerations](#) 22
[objects](#) 53
[overview](#) 22
[Path Bracket Record Types](#) 206

T

[Tracking changes](#) 302
[TransformRecordTypes packet](#) 231
[TriVertex Object](#) 80
[TriVertex packet](#) 80

U

[UniversalFontId Object](#) 81
[UniversalFontId packet](#) 81

V

[Vendor-extensible fields](#) 21
[Versioning](#) 20

W

[Weight enumeration](#) 51

X

[XForm Object](#) 82
[XForm packet](#) 82
[XHeight enumeration](#) 52