[MS-EMF-Diff]:

Enhanced Metafile Format

Revision Summary

Date	Revision History	Revision Class	Comments
4/3/2007	0.01	New	Version 0.01 release
7/3/2007	1.0	Major	MLonghorn+90
7/20/2007	2.0	Major	Restructured record sections according to category; other updates.
8/10/2007	2.1	Minor	Clarified the meaning of the technical content.
9/28/2007	2.2	Minor	Clarified the meaning of 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.
1/25/2008	3.2	Minor	Reconstructed record categories for clarity.
3/14/2008	4.0	Major	Abstract data model and Windows version-specific behavior added.
5/16/2008	4.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	4.1	Minor	Clarified the meaning of the technical content.
7/25/2008	4.1.1	Editorial	Changed language and formatting in the technical content.
8/29/2008	4.2	Minor	Clarified the meaning of the technical content.
10/24/2008	5.0	Major	Updated and revised the technical content.
12/5/2008	5.1	Minor	Clarified the meaning of the technical content.
1/16/2009	6.0	Major	Updated and revised the technical content.
2/27/2009	6.1	Minor	Clarified the meaning of the technical content.
4/10/2009	6.2	Minor	Clarified the meaning of the technical content.
5/22/2009	6.2.1	Editorial	Changed language and formatting in the technical content.
7/2/2009	6.3	Minor	Clarified the meaning of the technical content.
8/14/2009	6.4	Minor	Clarified the meaning of the technical content.
9/25/2009	6.5	Minor	Clarified the meaning of the technical content.
11/6/2009	6.5.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	6.6	Minor	Clarified the meaning of the technical content.
1/29/2010	6.6.1	Editorial	Changed language and formatting in the technical content.
3/12/2010	6.7	Minor	Clarified the meaning of the technical content.
4/23/2010	6.7.1	Editorial	Changed language and formatting in the technical content.
6/4/2010	6.8	Minor	Clarified the meaning of the technical content.
7/16/2010	6.8	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
8/27/2010	6.8	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	6.9	Minor	Clarified the meaning of the technical content.
11/19/2010	7.0	Major	Updated and revised the technical content.
1/7/2011	7.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	7.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	7.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	7.1	Minor	Clarified the meaning of the technical content.
9/23/2011	7.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	8.0	Major	Updated and revised the technical content.
3/30/2012	8.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	8.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	8.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	8.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	9.0	Major	Updated and revised the technical content.
11/14/2013	9.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	9.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	10.0	Major	Updated and revised the technical content.
6/30/2015	11.0	Major	Significantly changed the technical content.
10/16/2015	11.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	12.0	Major	Significantly changed the technical content.
6/1/2017	12.1	Minor	Clarified the meaning of the technical content.
9/15/2017	13.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The enhanced metafile format (EMF) is a file format that can store device-independent representations of graphics images. An EMF metafile can be parsed and processed to render the stored image on any output device.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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.

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. Each pixel has an alpha value that represents its level of transparency.

American National Standards Institute (ANSI) character set: A character set defined by a code page approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [ISO/IEC-8859-1]. In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-Unicode or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on Unicode, so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

anisotropic: Refers to the properties of an image, such as the scaling of logical units to device units, which are not the same regardless of the direction (x-axis versus y-axis) that is measured. Contrast with isotropic.

ASCII: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

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 two, which is 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.

big-endian: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

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

Boolean: An operation or expression that can be evaluated only as either true or false.

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

color gamut: The entire range of colors that is available on a particular graphics output device such as a display or printer.

color mapping: The process of associating integer color indices with color channel values.

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 plane: In bitmap graphics, all pixel information for a single color. See also color channel.

color profile: A file that contains information about how to convert colors in the color space and the color gamut of a specific device into a device-independent color space. A device-specific color profile is called a "device profile". For more information on using color and device profiles, see [MSDN-UDP].

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, 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.

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

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 font axes of a multiple master font.

device context: A collection of properties and objects that defines a dynamic environment for processes on a device. For graphics output, properties include brush style, line style, text layout, foreground and background colors, and mapping mode; and objects include a brush, pen, font, palette, region, and transform matrix. Multiple device contexts can exist simultaneously, but a single device context specifies the environment for graphics output at a particular point in time.

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.

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.

dithering: A form of digital halftoning.

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. Although EPS data can describe text, graphics, and images; the primary purpose of an EPS file is to be encapsulated within another PostScript page definition.

enhanced metafile format (EMF): A file format that supports the device-independent definitions of images.

enhanced metafile format plus extensions (EMF+): A file format that supports the device-independent definitions of images.

enhanced metafile spool format (EMFSPOOL): A format that specifies a structure of enhanced metafile format (EMF) records used for defining application and device-independent printer spool files.

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

font 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 correction: In digital imaging, the process of changing the brightness, contrast, or color balance of an image by assigning new values (different colors) to gray or color tones.

Graphics Device Interface (GDI): An API, supported on 16-bit and 32-bit versions of the operating system, that supports graphics operations and image manipulation on logical graphics objects.

Graphics Device Interface, Extended (GDI+): A Windows API, supported on 32-bit and 64-bit versions of the operating system, that extends GDI to include support for Bezier curves, gradient brushes, image effects, and EMF+ metafiles.

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

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

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

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 can occur in this area.

isotropic: Refers to the properties of an image, such as the scaling of logical units to device units, which are the same regardless of the direction (x-axis versus y-axis) that is measured. Contrast with anisotropic.

Joint Photographic Experts Group (JPEG): A raster graphics file format for displaying high-resolution color graphics. JPEG graphics apply a user-specified compression scheme that can significantly reduce the file sizes of photo-realistic color graphics. A higher level of compression results in lower quality, whereas a lower level of compression results in higher quality. JPEG-format files have a .jpg or .jpeg file name extension.

line cap: The shape that is used 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.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

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 is mapped to the system palette entry in order for the custom colors to appear when the application is run.

mapping mode: The way in which logical (device-independent) coordinates are mapped to device space (device-specific) coordinates. It also specifies the orientation of the axes and size of the units used for drawing operations.

metafile: A sequence of record structures that store an image in an application-independent format. Metafile records contain drawing commands, object definitions, and configuration settings. When a metafile is processed, the stored image can be rendered on a display, output to 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. Because multiple master fonts are outline fonts, 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 Microsoft implementation of OpenGL for the Windows operating system provides industry-standard graphics software for creating high-quality still and animated three-dimensional color images. See [OPENGL] for further information.

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.

original equipment manufacturer (OEM) character set: A character encoding used where the mappings between characters is dependent upon the code page configured on the machine, typically by the manufacturer.

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

page space: A logical coordinate system used for graphics operations. It is determined by the mapping mode. Page space is defined with device-independent units, such as pixels.

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

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 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 a metafile can be device-independent, playback is always associated with an output device with specific properties, such as resolution, color support, and so on.

Portable Network Graphics (PNG): A bitmap graphics file format that uses lossless data compression and supports variable transparency of images (alpha channels) and control of image brightness on different computers (gamma correction). PNG-format files have a .png file name extension.

PostScript: A page description language developed by Adobe Systems that is primarily used for printing documents on laser printers. It is the standard for desktop publishing.

print job: The rendered page description language (PDL) output data sent to a print device for a particular application or user request.

print server: A machine that hosts the print system and all its different components.

printer driver: The interface component between the operating system and the printer device. It is responsible for processing the application data into a page description language (PDL) that can be interpreted by the printer device.

raster operation: The process of combining the bits in a source bitmap with the bits in a destination bitmap and in a specified pattern, to achieve a particular graphical output.

rasterized font: A font produced with rasterization. Such fonts are not scalable; they 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. When magnified, the visual quality of a rasterized font decreases significantly compared to a vector font.

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

red-green-blue (RGB): A color model that describes color information in terms of the red (R), green (G), and blue (B) intensities in a color.

red-green-blue-alpha (RGBA): A color model that describes color information in terms of the red (R), green (G), blue (B), and alpha (A) intensities that comprise a color.

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.

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

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 [MS-EMF] section 2.1.31. Stock objects are neither created nor deleted.

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

tint: The amount of a neutral color, such as black or white, that is mixed with another color. Changing the tint increases or decreases the lightness and saturation, and leaves the hue 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.

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.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The Unicode standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

UTF-16LE (Unicode Transformation Format, 16-bits, little-endian): The encoding scheme specified in [UNICODE5.0.0/2007] section 2.6 for encoding Unicode characters as a sequence of 16-bit codes, each encoded as two 8-bit bytes with the least-significant byte first.

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

Windows Color System (WCS): Color management technology that ensures a color image, graphic, or text object is rendered as closely as possible to its original intent on any device, despite differences in imaging technologies and color capabilities between devices. WCS is a superset of ICM APIs and functionality and includes a variety of new functions that provide significant improvements in flexibility, transparency, predictability, and extensibility for vendors. Windows NT 3.1 operating system Windows NT 3.5 operating system, Windows NT 3.51 operating system, Windows 98 operating system, and Windows Millennium Edition operating system do not support WCS color management.

Windows metafile format (WMF): A file format used by Windows that supports the definition of images, including a format for clip art in word-processing documents.

world space: The most abstract logical coordinate space for graphics transforms. It allows scaling, translation, rotation, shearing, and reflection.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

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

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

[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, "The Unicode Consortium Home Page", http://www.unicode.org/

1.2.2 Informative References

[DEVMODE] Microsoft Corporation, "DEVMODE structure", http://msdn.microsoft.com/en-us/library/dd183565(VS.85).aspx

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

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

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

[MSDN-WindowsGDI] Microsoft Corporation, "Windows GDI", http://msdn.microsoft.com/en-us/library/dd145203.aspxx

[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

An EMF metafile consists of an ordered sequence of variable-length records (section 2.1.1) that contain drawing commands, object definitions, and graphics 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 output device.

The image defined in an EMF metafile maintains its dimensions, shape, and proportions on any output device, including printers, plotters, and monitors, or in the client areas of applications.

1.3.1 Metafile Structure

The first record in an EMF metafile is a header (section 2.3.4.2), and the last is an end-of-file (section 2.3.4.1). Between them are records that specify the rendering of the image.

The following diagram depicts the different versions of EMF metafiles, which are:

- **Original**: The original version of EMF defined a file container for the device-independent specification of images.<1>
- **Extension 1**: The second version and first extension added a pixel format record and support for OpenGL commands, enhancing the device independence and flexibility of EMF.
- **Extension 2**: The third version and second extension added the capability to measure distances on device surfaces by using the Metric system, enhancing the accuracy and scalability of EMF.

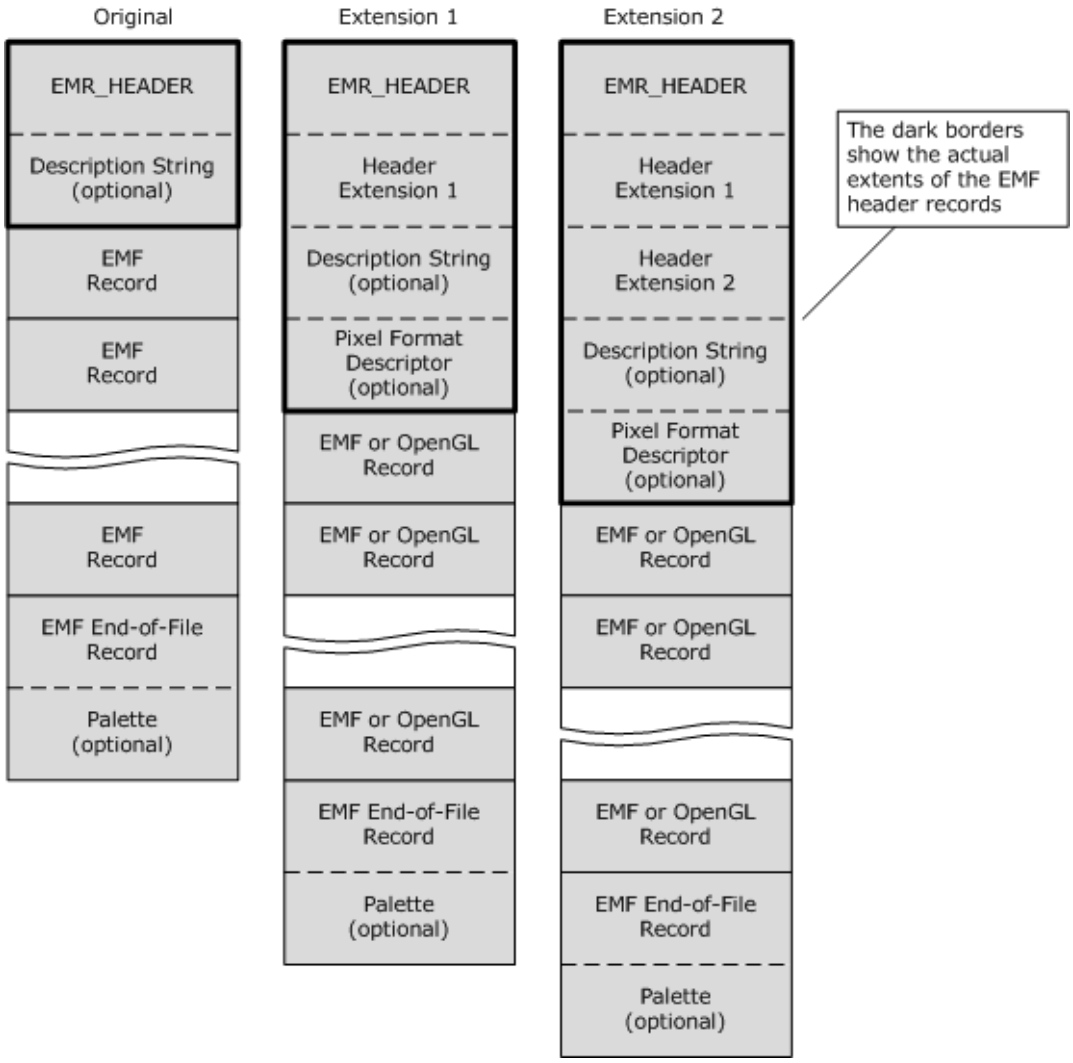


Figure 1: EMF metafile versions

EMF metafiles can be considered to have three sections:

- **EMF header**: The header record (section 2.3.4.2), possibly with extensions. It contains information concerning the structure and contents of the metafile, including an optional description string and pixel format descriptor.
- **EMF records:** A sequence of drawing orders, property settings, and object definitions (section 2.3). At least one record is present, not counting the header and end-of-file records.

- **EMF end-of-file**: The end-of-file record (section 2.3.4.1), which is the last record in the EMF metafile.

EMF records are contiguous because the information that is available for traversing the metafile from record to record depends on it. From any given EMF record, except the end-of-file, the length of that record can be used to move to the next record in the metafile.

1.3.2 Graphics Objects

Graphics objects are used by the drawing and painting operations specified in EMF metafile records. They are created by object creation records (section 2.3.7) and saved in an object table (section 3.1.1.1). When needed, a graphics object can be selected into the playback device context. by using an EMR_SELECTOBJECT record (section 2.3.8.5).

The types of reusable objects include:

- Brushes (2.2.12)
- Color spaces ([MS-WMF] sections 2.2.2.11 and 2.2.2.12)
- Fonts (sections 2.2.13, 2.2.14, 2.2.15, and 2.2.16)
- Palettes (section 2.2.17)
- Pens (sections 2.2.19 and 2.2.20)

Stock objects are used until graphics objects are created.

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

For an example of how the use of the big-endian and little-endian methods can affect the compatibility of applications, see section 3.1.2.

1.4 Relationship to Protocols and Other Structures

EMF is related to the following metafile formats:

- Enhanced metafile spool format (EMFSPOOL) [MS-EMFSPOOL]: EMFSPOOL metafiles can contain EMF metafiles. They are used for print job spooling.
- Enhanced metafile format plus extensions (EMF+) [MS-EMFPLUS]: EMF+ metafiles can be embedded in EMF metafiles.<2>

- Windows metafile format (WMF) [MS-WMF]: WMF metafiles can be embedded in EMF+ metafiles. WMF graphics are not device-independent.

This relationship is illustrated by the following figure:

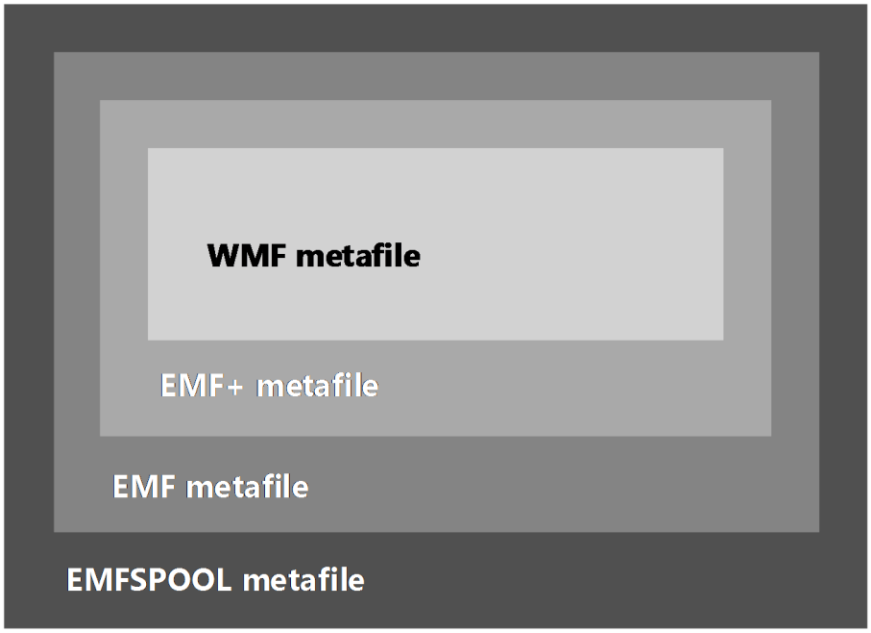


Figure 2: Relationship to other metafile formats

1.5 Applicability Statement

Files that adhere to the EMF metafile format are portable, device-independent containers for images. The graphics supported in EMF metafiles are applicable to all representations of document content on all devices, including displays, printers, and plotters.

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.
- Extension 2: Added the capability of measuring display dimensions in micrometers.

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 EMR_COMMENT record (section 2.3.3.1) 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 EMF enumerations, objects, and records.

All character strings specified in this section are encoded in Unicode UTF16-LE format [UNICODE] unless stated otherwise.

Unreferenced intrinsic types are defined in [MS-DTYP].

2.1 EMF Enumerations

2.1.1 RecordType Enumeration

The **RecordType** enumeration defines values that uniquely identify records in an EMF metafile. These values are specified in the **Type** fields of EMF records (section 2.3).

```
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_SETPOLYFILLMODE = 0x00000013,
  EMR_SETROP2 = 0x00000014,
  EMR_SETSTRETCHBLTMODE = 0x00000015,
  EMR_SETTEXTALIGN = 0x00000016,
  EMR_SETCOLORADJUSTMENT = 0x00000017,
  EMR_SETTEXTCOLOR = 0x00000018,
  EMR_SETBKCOLOR = 0x00000019,
  EMR_OFFSETCLIPRGN = 0x0000001A,
  EMR_MOVETOEX = 0x0000001B,
  EMR_SETMETARGN = 0x0000001C,
  EMR_EXCLUDECLIPRECT = 0x0000001D,
  EMR_INTERSECTCLIPRECT = 0x0000001E,
  EMR_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_ROUNDRECT = 0x0000002C,
  EMR_ARC = 0x0000002D,
  EMR_CHORD = 0x0000002E,
```

```
EMR_PIE = 0x0000002F,
EMR_SELECTPALETTE = 0x00000030,
EMR_CREATEPALETTE = 0x00000031,
EMR_SETPALETTEENTRIES = 0x00000032,
EMR_RESIZEPALETTE = 0x00000033,
EMR_REALIZEPALETTE = 0x00000034,
EMR_EXTFLOODFILL = 0x00000035,
EMR_LINETO = 0x00000036,
EMR_ARCTO = 0x00000037,
EMR_POLYDRAW = 0x00000038,
EMR_SETARCDIRECTION = 0x00000039,
EMR_SETMITERLIMIT = 0x0000003A,
EMR_BEGINPATH = 0x0000003B,
EMR_ENDPATH = 0x0000003C,
EMR_CLOSEFIGURE = 0x0000003D,
EMR_FILLPATH = 0x0000003E,
EMR_STROKEANDFILLPATH = 0x0000003F,
EMR_STROKEPATH = 0x00000040,
EMR_FLATTENPATH = 0x00000041,
EMR_WIDENPATH = 0x00000042,
EMR_SELECTCLIPPATH = 0x00000043,
EMR_ABORTPATH = 0x00000044,
EMR_COMMENT = 0x00000046,
EMR_FILLRGN = 0x00000047,
EMR_FRAMERGN = 0x00000048,
EMR_INVERTRGN = 0x00000049,
EMR_PAINTRGN = 0x0000004A,
EMR_EXTSELECTCLIPRGN = 0x0000004B,
EMR_BITBLT = 0x0000004C,
EMR_STRETCHBLT = 0x0000004D,
EMR_MASKBLT = 0x0000004E,
EMR_PLGBLT = 0x0000004F,
EMR_SETDIBITSTODEVICE = 0x00000050,
EMR_STRETCHDIBITS = 0x00000051,
EMR_EXTCREATEFONTINDIRECTW = 0x00000052,
EMR_EXTTEXTOUTA = 0x00000053,
EMR_EXTTEXTOUTW = 0x00000054,
EMR_POLYBEZIER16 = 0x00000055,
EMR_POLYGON16 = 0x00000056,
EMR_POLYLINE16 = 0x00000057,
EMR_POLYBEZIERTO16 = 0x00000058,
EMR_POLYLINETO16 = 0x00000059,
EMR_POLYPOLYLINE16 = 0x0000005A,
EMR_POLYPOLYGON16 = 0x0000005B,
EMR_POLYDRAW16 = 0x0000005C,
EMR_CREATEMONOBRUSH = 0x0000005D,
EMR_CREATEDIBPATTERNBRUSHPT = 0x0000005E,
EMR_EXTCREATEPEN = 0x0000005F,
EMR_POLYTEXTOUTA = 0x00000060,
EMR_POLYTEXTOUTW = 0x00000061,
EMR_SETICMMODE = 0x00000062,
EMR_CREATECOLORSPACE = 0x00000063,
EMR_SETCOLORSPACE = 0x00000064,
EMR_DELETECOLORSPACE = 0x00000065,
EMR_GLSRECORD = 0x00000066,
EMR_GLSBOUNDEDRECORD = 0x00000067,
EMR_PIXELFORMAT = 0x00000068,
EMR_DRAWESCAPE = 0x00000069,
EMR_EXTESCAPE = 0x0000006A,
EMR_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 drawing position.

EMR_POLYLINETO: This record defines one or more straight lines based upon the current drawing position. A line is drawn from the current drawing 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_SETVIEWPORTORGEX: This record defines the viewport origin.

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

EMR_EOF: This record indicates the end of the metafile.

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

EMR_SETMAPPERFLAGS: This record specifies parameters for the process of matching logical fonts to physical fonts, which is performed by the font mapper.

EMR_SETMAPMODE: This record defines the mapping mode, which defines the unit of measure used to transform page space units into device space units, and defines the orientation of the device's X and Y axes.

EMR_SETBKMODE: This record defines the background mix mode, which 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 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 current clipping region by the specified offsets.

EMR_MOVETOEX: This record defines coordinates of the new drawing position in logical units.

EMR_SETMETARGN: This record intersects the current clipping region with the current metaregion and saves the combined region as the new current metaregion.

EMR_EXCLUDECLIPRECT: This record defines a new clipping region that consists of the current clipping region intersected with 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 using the ratios formed by the specified multiplicands and divisors.

EMR_SCALEWINDOWEXTEX: This record redefines the window using the ratios formed by the specified multiplicands and divisors.

EMR_SAVEDC: This record saves the current state of the playback device context (section 3.1) in an array of states saved by preceding **EMR_SAVEDC** records if any.

EMR_RESTOREDC: This record restores the playback device context to the specified state, which was saved by a preceding **EMR_SAVEDC** record (section 2.3.11).

EMR_SETWORLDTRANSFORM: This record defines a two-dimensional linear transform between world space and page space [MSDN-WRLDPGSPC].

EMR_MODIFYWORLDTRANSFORM: This record redefines the world transform by using the specified mode.

EMR_SELECTOBJECT: This record selects an object in the playback device context, which is identified by its index in the EMF object table (section 3.1.1.1).

EMR_CREATEPEN: This record defines a logical pen (section 2.2.19) that has the specified style, width, and color.

EMR_CREATEBRUSHINDIRECT: This record defines a logical brush for filling figures in graphics operations.

EMR_DELETEOBJECT: This record deletes a graphics object, clearing its index in the EMF object table.

EMR_ANGLEARC: This record defines a line segment of an arc. The line segment is drawn from the current drawing 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, 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.

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 selects a LogPalette object (section 2.2.17) into 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 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 drawing position up to, but not including, the specified point. It resets the current drawing position to the specified point.

EMR_ARCTO: This record defines an elliptical arc. It resets the current position to the endpoint 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.

EMR_BEGINPATH: This record opens a path bracket for specifying the current path.

EMR_ENDPATH: This record closes an open path bracket and selects the path 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 bracket and fills its 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 turns each curve in the path into a sequence of lines.

EMR_WIDENPATH: This record redefines the current path bracket as the area that would be painted if the path were stroked using the current pen.

EMR_SELECTCLIPPATH: This record specifies a clipping region as the current clipping region combined with the current path bracket, 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 current brush.

EMR_EXTSELECTCLIPRGN: This record combines the specified region with the current clipping 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.

EMR_EXTTEXTOUTA: This record draws an ASCII text string using the current font and text colors.

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

EMR_POLYLINETO16: This record defines one or more straight lines based upon the current position. A line is drawn from the current position to the first point specified by the **Points** field by using the current pen. For each additional line, drawing is performed from the ending point of the previous line to the next point specified by **Points**.

EMR_POLYPOLYLINE16: This record defines multiple series of connected line segments.

EMR_POLYPOLYGON16: This record defines a series of closed polygons. Each polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygons specified by this record can overlap.

EMR_POLYDRAW16: This record defines a set of line segments and Bezier curves.

EMR_CREATEMONOBRUSH: This record defines a logical brush with the specified bitmap pattern. The bitmap can be a 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 an extended logical pen (section 2.2.20) that has the specified style, width, color, 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_EXTTEXTOUTW records, one per string.<3>

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_EXTTEXTOUTW records, one per string.<4>

EMR_SETICMMODE: This record specifies the mode of Image Color Management (ICM) for graphics operations.<5>

EMR_CREATECOLORSPACE: This record creates a logical color space object from a color profile with a name consisting of ASCII characters.<6>

EMR_SETCOLORSPACE: This record defines the current logical color space object for graphics operations.<7>

EMR_DELETECOLORSPACE: This record deletes a logical color space object.<8>

Note: An EMR_DELETEOBJECT record SHOULD be used instead of EMR_DELETECOLORSPACE to delete a logical color space object.<9>

EMR_GLSRECORD: This record specifies an OpenGL function.<10>

EMR_GLSBOUNDEDRECORD: This record specifies an OpenGL function with a bounding rectangle for output.<11>

EMR_PIXELFORMAT: This record specifies the pixel format to use for graphics operations.<12>

EMR_DRAWESCAPE: This record passes arbitrary information to the driver. The intent is that the information results in drawing being done.

EMR_EXTESCAPE: This record passes arbitrary information to the driver. The intent is that the information does not result in drawing being done.

EMR_SMALLTEXTOUT: This record outputs a string.

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

EMR_NAMEDESCAPE: This record passes arbitrary information to the given named driver.

EMR_COLORCORRECTPALETTE: This record specifies how to correct the entries of a logical palette object using Windows Color System (WCS) 1.0 values.<13>

EMR_SETICMPROFILEA: This record specifies a color profile in a file with a name consisting of ASCII characters, for graphics output.<14>

EMR_SETICMPROFILEW: This record specifies a color profile in a file with a name consisting of Unicode characters, for graphics output.<15>

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.<16>

EMR_SETLAYOUT: This record specifies the order in which text and graphics are drawn.<17>

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.<18>

EMR_GRADIENTFILL: This record specifies filling rectangles or triangles with gradients of color.<19>

EMR_SETLINKEDUFIS: This record sets the **UniversalFontIds** (section 2.2.27) 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.<20>

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.<21>

EMR_CREATECOLORSPACEW: This record creates a logical color space object from a color profile with a name consisting of Unicode characters.<22>

2.1.2 ArcDirection Enumeration

The **ArcDirection** enumeration is used in setting the drawing direction for arcs and rectangles.

```
typedef  enum
{
  AD_COUNTERCLOCKWISE = 0x00000001,
  AD_CLOCKWISE = 0x00000002
```

```
} 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 is 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 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 current color 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 [MS-WMF].

EMR_COMMENT_BEGINGROUP: This comment record identifies the beginning of a group of drawing records.

EMR_COMMENT_ENDGROUP: This comment record identifies the end of a group of drawing records.

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.

EMR_COMMENT_UNICODE_END: This comment record is reserved and MUST NOT be used.

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 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.<23>

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.<24>

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 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.<25>

ETO_NUMERICSLATIN: This bit indicates that to display numbers, European digits SHOULD be used.<26>

ETO_IGNORELANGUAGE: This bit indicates that no special operating system processing for glyph placement is performed on right-to-left strings; that is, all glyph positioning SHOULD be taken care of by drawing and state records in the metafile.<27>

ETO_PDY: This bit indicates that both horizontal and vertical character displacement values SHOULD be provided.<28>

ETO_REVERSE_INDEX_MAP: This bit is reserved and SHOULD NOT be used.<29>

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 metafiles.

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

ENHMETA_SIGNATURE: The sequence of ASCII characters "FME ", which denotes EMF data. The reverse of the string is " EMF".

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

This signature is used in the following structures:

- EMR_HEADER records (section 2.3.4.2) to identify the EMF metafile
- The EmrFormat object (section 2.2.4) in EMR_COMMENT_MULTIFORMATS records (section 2.3.3.4.3), to specify embedded EMF records.

EPS_SIGNATURE: The value of this member is the sequence of ASCII characters "FSPE", which denotes encapsulated PostScript (EPS) data. The reverse of the string is "EPSF".

This signature is used in EmrFormat objects to specify embedded PostScript data in the EpsData object (section 2.2.6) in EMR_COMMENT_MULTIFORMATS records.

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: Color interpolation along a gradient from the left to the right edges of a rectangle.

GRADIENT_FILL_RECT_V: Color interpolation along a gradient from the top to the bottom edges of a rectangle.

GRADIENT_FILL_TRIANGLE: Color interpolation 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 transform. Only the height of the text SHOULD be scaled.<30>

Arcs MUST be drawn using the current arc direction, but they MUST NOT reflect the current world-to-device transform, which might require a rotation along the x-axis or y-axis.

The world-to-device transform is 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 2.3.11.29) records, respectively.

The world-to-device transform can be changed by EMR_MODIFYWORLDTRANSFORM (section 2.3.12.1) and EMR_SETWORLDTRANSFORM (section 2.3.12.2) records.

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

GM_ADVANCED: TrueType text output SHOULD<31> fully conform to the current world-to-device transform.

Arcs MUST be drawn in the counterclockwise direction in world space; however, both arc control points and the arcs themselves MUST reflect the current world-to-device transform.

The world-to-device transform can be modified directly by EMR_MODIFYWORLDTRANSFORM and EMR_SETWORLDTRANSFORM records, or indirectly by changing the window and viewport extents and origins, using the EMR_SETWINDOWEXTEX and EMR_SETVIEWPORTEXTEX records, and the EMR_SETWINDOWORGEX and EMR_SETVIEWPORTORGEX 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 Image Color Management (ICM).<32>

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

ICM_OFF: Turns off ICM; turns on old-style color correction of halftones.

ICM_ON: Turns on ICM; turns off old-style color correction of halftones.

ICM_QUERY: Queries the current state of color management.

ICM_DONE_OUTSIDEDC: Turns off both ICM and old-style color correction of halftones.

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 isotropic; that is, they are mapped to arbitrary units with equally scaled axes. Thus, one unit along the x-axis is equal to one unit along the y-axis. The EMR_SETWINDOWEXTEX (section 2.3.11.30) and EMR_SETVIEWPORTEXTEX (section 2.3.11.28) records are 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 anisotropic; that is, they are mapped to arbitrary units with arbitrarily scaled axes. The EMR_SETWINDOWEXTEX and EMR_SETVIEWPORTEXTEX records are used to specify the units, orientation, and scaling of the axes.

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 changing 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 current transform is the right multiplicand.

MWT_RIGHTMULTIPLY: Multiply the current transform. In this mode, the specified transform data is the right multiplicand, and the current transform is the left multiplicand.

MWT_SET: Set the current transform to the specified transform data.

The transform data is specified as an XForm object (section 2.2.28).

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, considering 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. 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 (section 2.3.11.21).

PS_GEOMETRIC: 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 bracket or a new region that is being combined with the current clipping 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 bracket (or new region).

RGN_OR: The new clipping region includes the union (combined areas) of the current clipping region and the current path bracket (or new region).

RGN_XOR: The new clipping region includes the union of the current clipping region and the current path bracket (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 bracket (or new region) excluded.

RGN_COPY: The new clipping region is the current path bracket (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 from the BrushStyle enumeration ([MS-WMF] section 2.1.1.4)
- **Color**: 0x00FFFFFF in a 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:

- **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 from the PenStyle enumeration (section 2.1.25)
- **ColorRef**: 0x00FFFFFF in a 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 LogFont object (section 2.2.13) with the following properties:

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

ANSI_FIXED_FONT: A fixed-width font that is equivalent to a LogFont object with the following properties:<33>

- **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:<34>

- **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.<35>

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.<36>

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.<37>

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.<38>

DC_BRUSH: The solid-color brush that is currently selected in the playback device context. The default SHOULD<39> be WHITE_BRUSH.

DC_PEN: The solid-color pen that is currently selected in the playback device context. The default SHOULD<40> be BLACK_PEN.

During metafile processing, stock object indexes can be used by object manipulation records (section 2.3.8) 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.<41>

```
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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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 real 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.<42>

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

Size (2 bytes): An unsigned integer that specifies the size in bytes of this object. This value is 0x0018.

Values (2 bytes): An 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): An 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): An 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.<43> A value of 10,000 means gamma correction MUST NOT be performed.

GreenGamma (2 bytes): An 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): An 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): An 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): An 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 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 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 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.

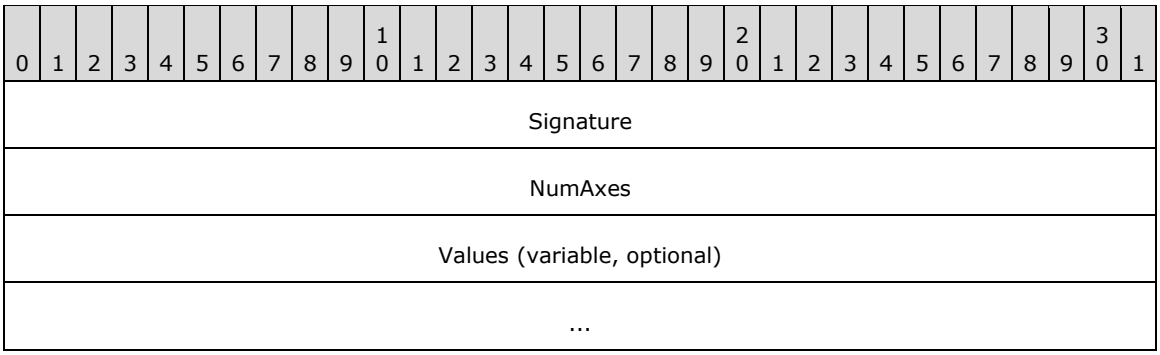
RedGreenTint (2 bytes): A signed integer that specifies the amount of red or green tint adjustment to be applied to the source object. This value SHOULD be in the range from –100 to 100. Positive numbers adjust towards red and negative numbers adjust towards green. A value of zero means tint adjustment 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.



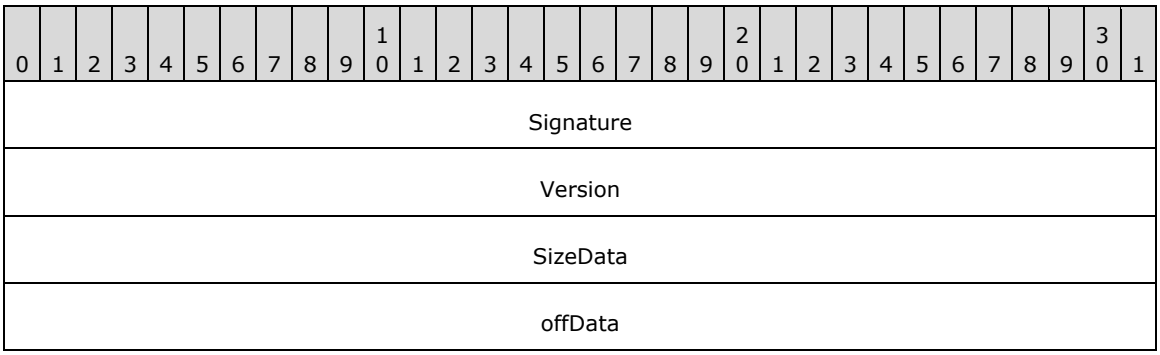
Signature (4 bytes): An unsigned integer that MUST be set to the value 0x08007664.

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

Values (variable, optional): An 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).



Signature (4 bytes): An unsigned integer that specifies the format of the image data. This value is in the FormatSignature enumeration (section 2.1.14).

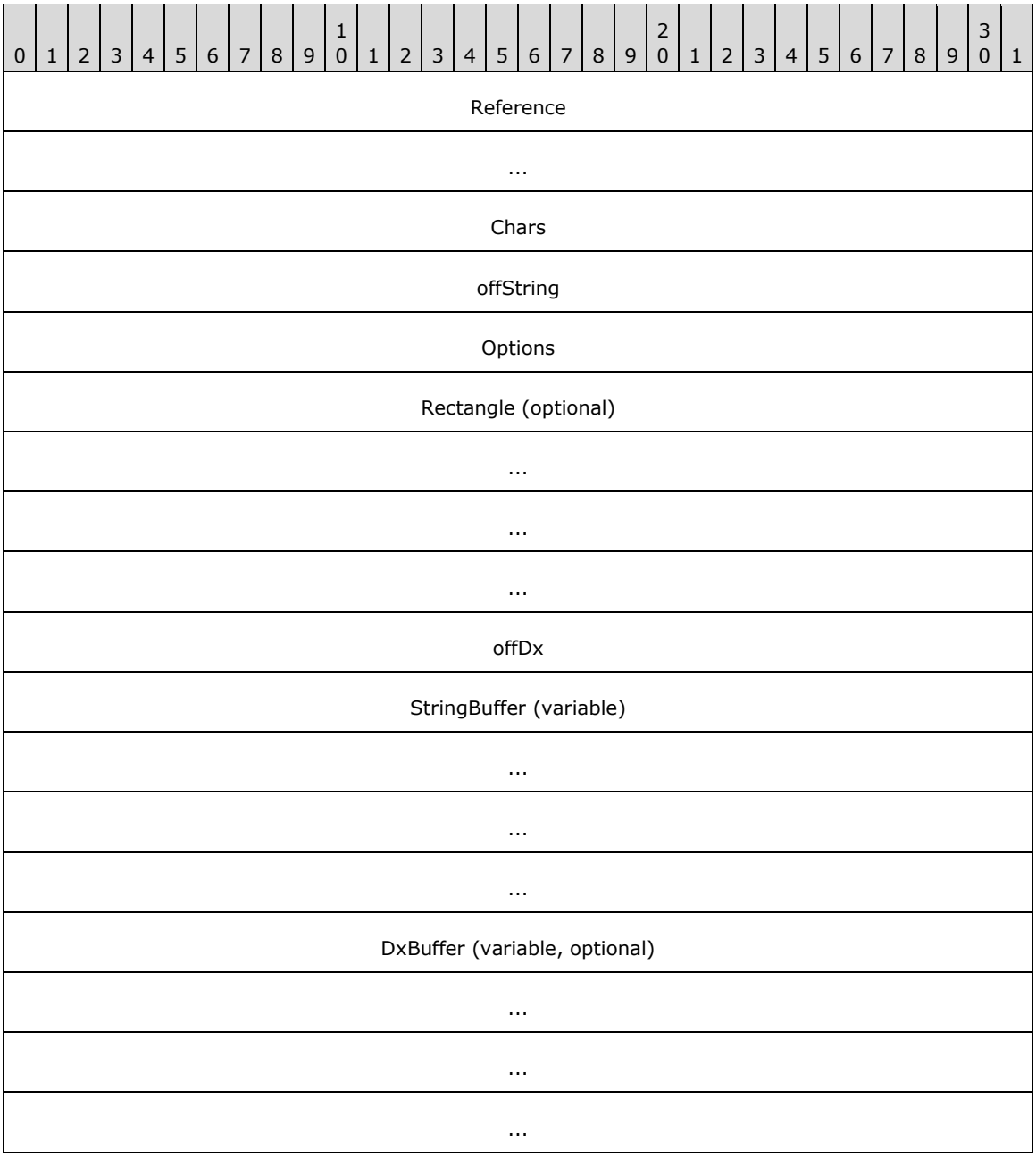
Version (4 bytes): An unsigned integer that specifies the format version number. If the **Signature** field specifies encapsulated PostScript (EPS), this value is 0x00000001; otherwise, this value is ignored.

SizeData (4 bytes): An unsigned integer that specifies the size of the data in bytes.

offData (4 bytes): An 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.



Reference (8 bytes): A 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), which is specified using TextAlignmentMode flags ([MS-WMF] section 2.1.2.3).

Chars (4 bytes): An unsigned integer that specifies the number of characters in the string.

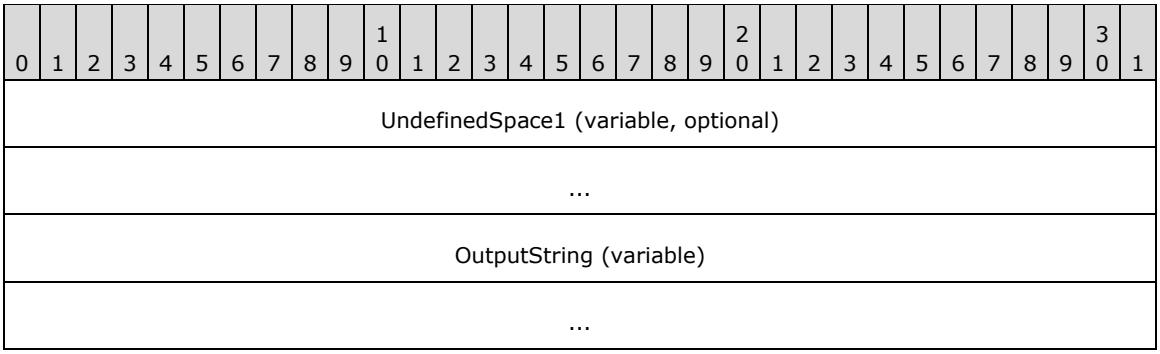
offString (4 bytes): An unsigned integer that specifies the offset to the output string in bytes, from the start of the record in which this object is contained. This value is 8- or 16-bit aligned, according to the character format.

Options (4 bytes): An 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, optional): A 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.<44>

offDx (4 bytes): An 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 is 32-bit aligned.

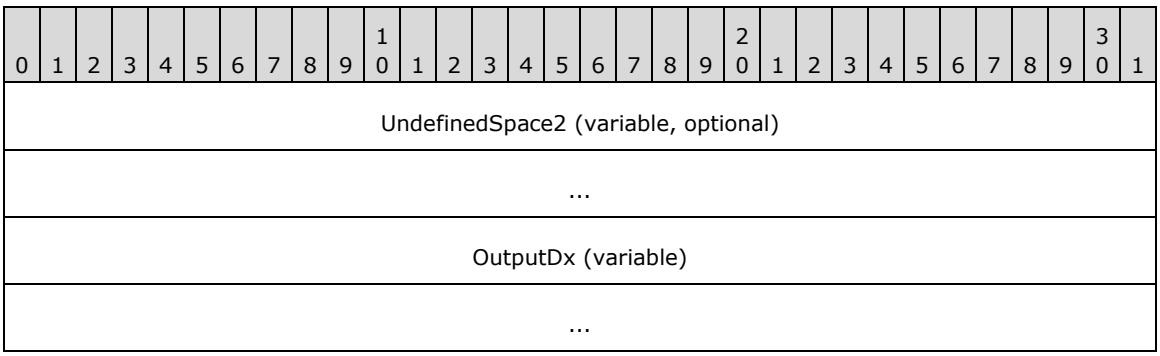
StringBuffer (variable): The character string buffer.



UndefinedSpace1 (variable, optional): The 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, optional): The character spacing buffer.



UndefinedSpace2 (variable, optional): The 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.

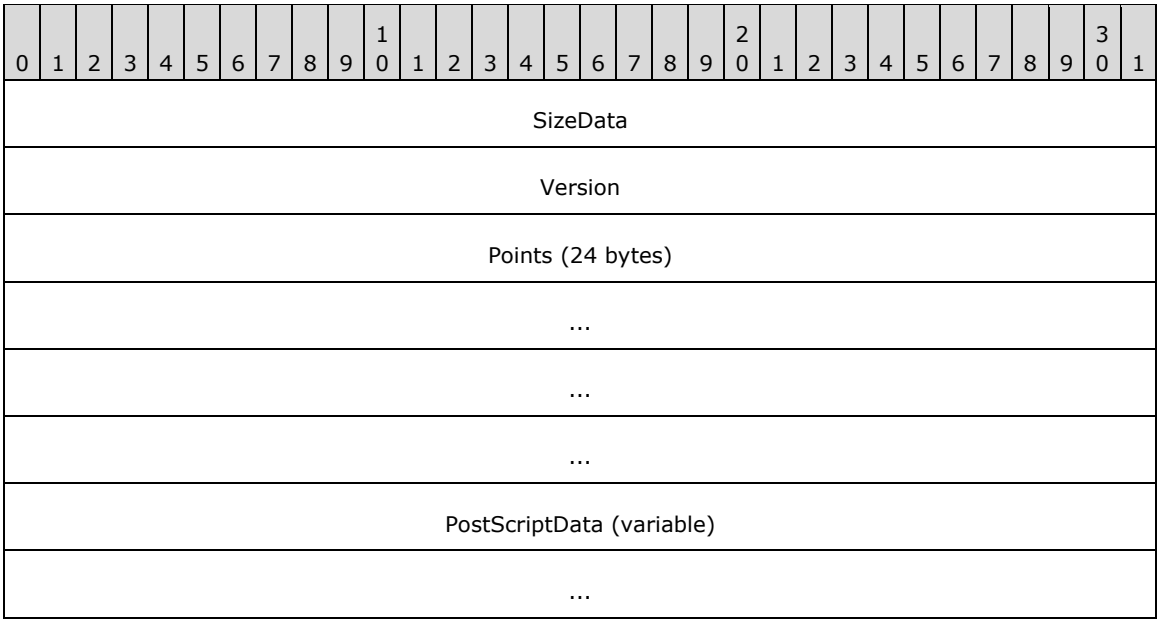
If ETO_RTLREADING is specified, characters are laid right to left instead of left to right. No other options affect the interpretation of this field.

The size and encoding of the characters in the **OutputString** is determined by the type of record that contains this 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 (section 2.3.5.8) and EMR_POLYTEXTOUTW (section 2.3.5.33) records: 16-bit Unicode UTF16-LE characters.

2.2.6 EpsData Object

The EpsData object is a container for EPS data.



SizeData (4 bytes): An unsigned integer that specifies the total size of this object in bytes.

Version (4 bytes): An unsigned integer that specifies the PostScript language level. This value is 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
```

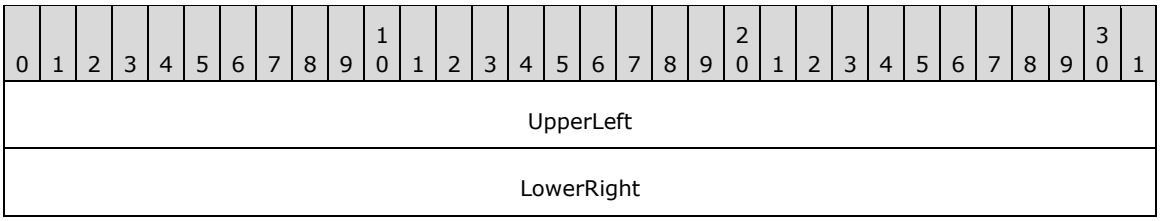
PostScriptData (variable): An array of bytes of PostScript data. The length of this array can be computed from the **SizeData** field. This data MAY be used to render an image.<45>

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).

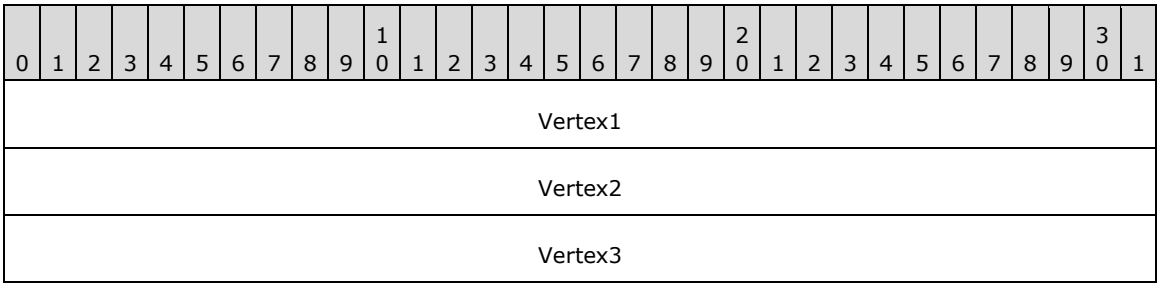


UpperLeft (4 bytes): An index into an array of TriVertex objects that specifies the upper-left vertex of a rectangle. The index MUST be smaller than the size of the array, as defined by the **nVer** field of the EMR_GRADIENTFILL record.

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).



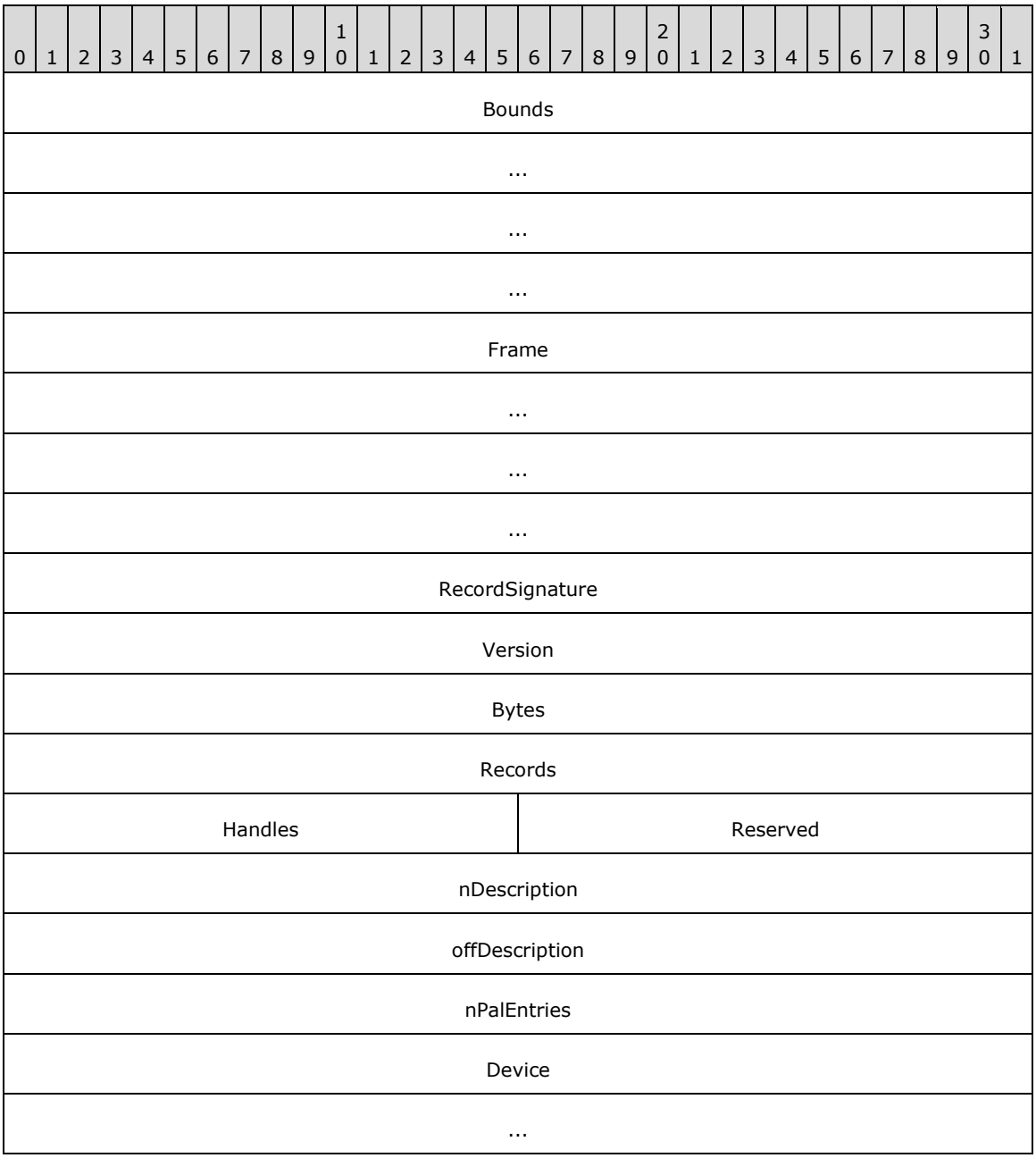
Vertex1 (4 bytes): An index into an array of TriVertex objects that specifies a vertex of a triangle. The index MUST be smaller than the size of the array, as defined by the **nVer** field of the EMR_GRADIENTFILL record.

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.



Millimeters
...

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the rectangular inclusive-inclusive bounds in logical units of the smallest rectangle that can be drawn around the image stored in the metafile.

Frame (16 bytes): A 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): An unsigned integer that specifies the record signature. This MUST be **ENHMETA_SIGNATURE**, from the FormatSignature enumeration (section 2.1.14).

Version (4 bytes): An unsigned integer that specifies the EMF version for interoperability. This MAY be 0x00010000.

Bytes (4 bytes): An unsigned integer that specifies the size of the metafile in bytes.

Records (4 bytes): An unsigned integer that specifies the number of records in the metafile.

Handles (2 bytes): An unsigned integer that specifies the number of graphics objects that are used during the processing of the metafile.

Reserved (2 bytes): An unsigned integer that MUST be 0x0000 and MUST be ignored.

nDescription (4 bytes): An 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): An 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): An 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 SizeL object ([MS-WMF] section 2.2.2.22) that specifies the size of the reference device, in pixels.

Millimeters (8 bytes): A 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cbPixelFormat																															
offPixelFormat																															
bOpenGL																															

cbPixelFormat (4 bytes): An unsigned integer that specifies the size of the PixelFormatDescriptor object. This value is 0x00000000 if no pixel format is set.

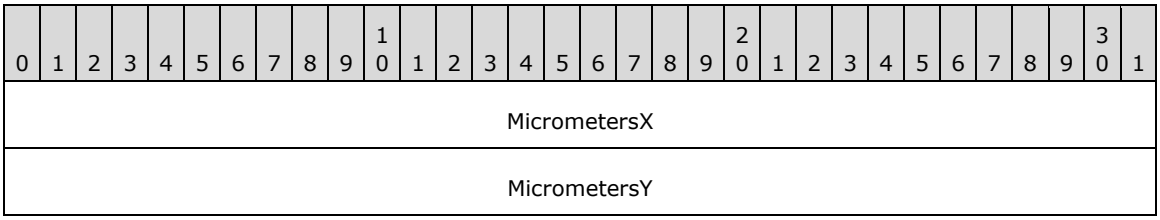
offPixelFormat (4 bytes): An unsigned integer that specifies the offset to the PixelFormatDescriptor object. This value is 0x00000000 if no pixel format is set.

bOpenGL (4 bytes): An 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.

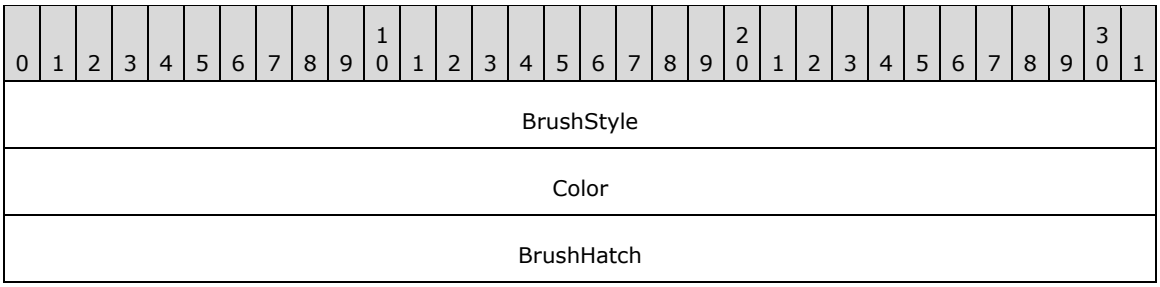


MicrometersX (4 bytes): The 32-bit horizontal size of the display device for which the metafile image was generated, in micrometers.

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): An unsigned integer that specifies the brush style. The value MUST be an enumeration from 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.

Color (4 bytes): A 32-bit 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	A 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	A ColorRef object, which specifies the foreground color of the hatch pattern.	A value from the HatchStyle enumeration (section 2.1.17), 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.

<table>
<tr><td colspan="32">0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1</td></tr>
<tr><td colspan="32">Height</td></tr>
<tr><td colspan="32">Width</td></tr>
<tr><td colspan="32">Escapement</td></tr>
<tr><td colspan="32">Orientation</td></tr>
<tr><td colspan="32">Weight</td></tr>
<tr><td colspan="8">Italic</td><td colspan="8">Underline</td><td colspan="8">StrikeOut</td><td colspan="8">CharSet</td></tr>
<tr><td colspan="8">OutPrecision</td><td colspan="8">ClipPrecision</td><td colspan="8">Quality</td><td colspan="8">PitchAndFamily</td></tr>
<tr><td colspan="32">Facename (64 bytes)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Height (4 bytes): A signed integer that specifies the height of the font's character cell in logical units. 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 < 0x00000000	The font mapper transforms this value into device units and matches its absolute value against the character height of the available fonts.

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

Width (4 bytes): A signed integer that specifies the average width of characters in the font in logical units. If the **Width** field value is zero, an appropriate value SHOULD<46> be calculated from other values in this object to find a font that has the typographer's intended aspect ratio.

Escapement (4 bytes): A 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 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 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.<47>

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

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

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

CharSet (1 byte): An unsigned integer that specifies the set of character glyphs. It MUST be a value in the 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 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 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 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 ClipPrecision Flags ([MS-WMF] section 2.1.2.1).

Quality (1 byte): An 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 FontQuality enumeration ([MS-WMF] section 2.1.1.10).

PitchAndFamily (1 byte): A 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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
LogFont (92 bytes)																															
...																															
...																															
...																															
FullName (128 bytes)																															
...																															
...																															
...																															
Style (64 bytes)																															
...																															
...																															
...																															
Script (64 bytes)																															
...																															
...																															

...

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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
LogFontEx (348 bytes)																															
...																															
...																															
...																															
DesignVector (variable)																															
...																															

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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
LogFont (92 bytes)																															
...																															

...	
...	
FullName (128 bytes)	
...	
...	
...	
Style (64 bytes)	
...	
...	
...	
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): An 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): An unsigned integer that MUST be set to zero and MUST be ignored.

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

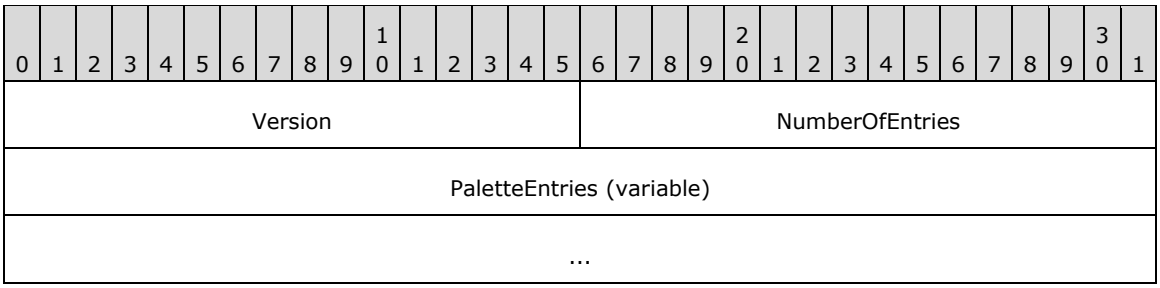
Culture (4 bytes): An unsigned integer that MUST be set to zero and MUST be ignored.

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.



Version (2 bytes): An unsigned integer that specifies the version number of the system. This value is 0x0300.

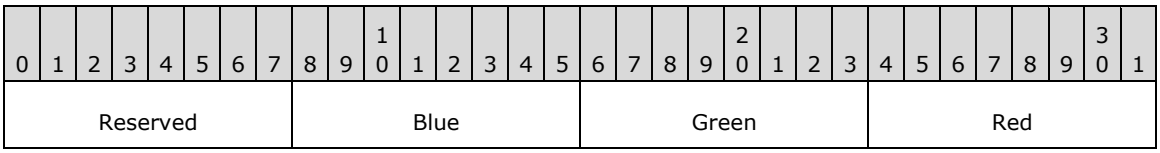
NumberOfEntries (2 bytes): An unsigned integer that specifies the number of entries in the **PaletteEntries** field.

PaletteEntries (variable): An array of LogPaletteEntry objects (section 2.2.18) 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).



Reserved (1 byte): An unsigned integer that MUST NOT be used and MUST be ignored.

Blue (1 byte): An unsigned integer that defines the blue intensity value for the entry.

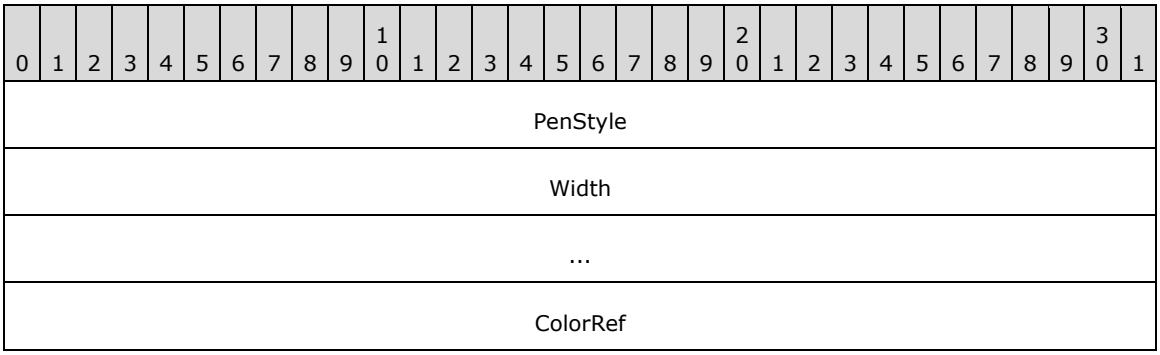
Green (1 byte): An unsigned integer that defines the green intensity value for the entry.

Red (1 byte): An unsigned integer that defines the red intensity value for the 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.



PenStyle (4 bytes): An unsigned integer that specifies a value from the PenStyle enumeration (section 2.1.25).

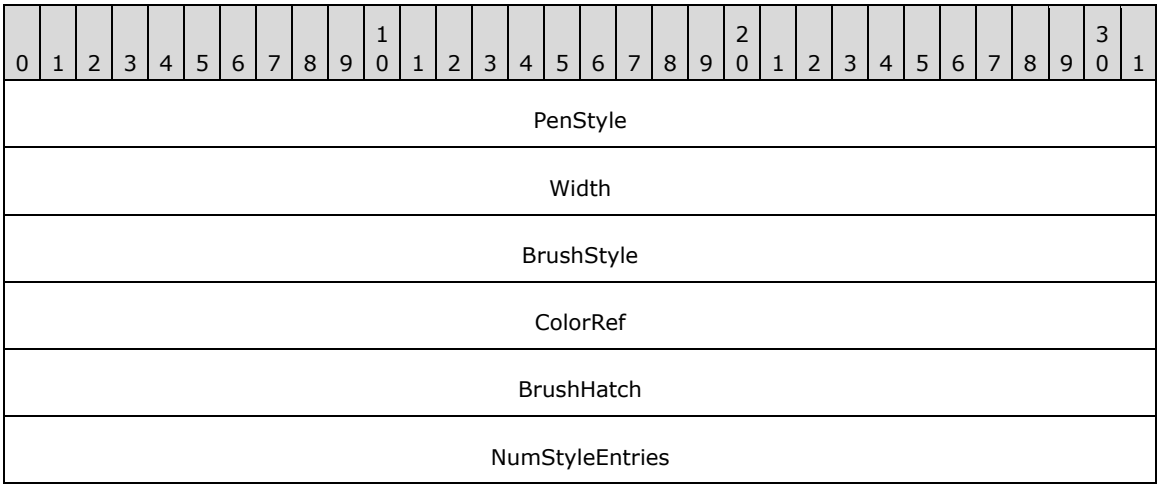
Width (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15) that specifies the width of the pen by the value of its **x** field. The value of its **y** field MUST be ignored.

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 **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.



StyleEntry (variable, optional)
...

PenStyle (4 bytes): An unsigned integer that specifies the pen style. This value is defined from the PenStyle enumeration (section 2.1.25).

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

Width (4 bytes): An 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): An unsigned integer that specifies a brush style for the pen from the BrushStyle enumeration ([MS-WMF] section 2.1.1.4).

If the pen type in the **PenStyle** field is PS_GEOMETRIC, this value is 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.

ColorRef (4 bytes): A 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, optional): An 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. The array contains the number of entries specified by **NumStyleEntries**, but it is used as if it repeated indefinitely.

The first entry in the **StyleEntry** 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, lengths are specified in logical units; otherwise, they are specified in device units.

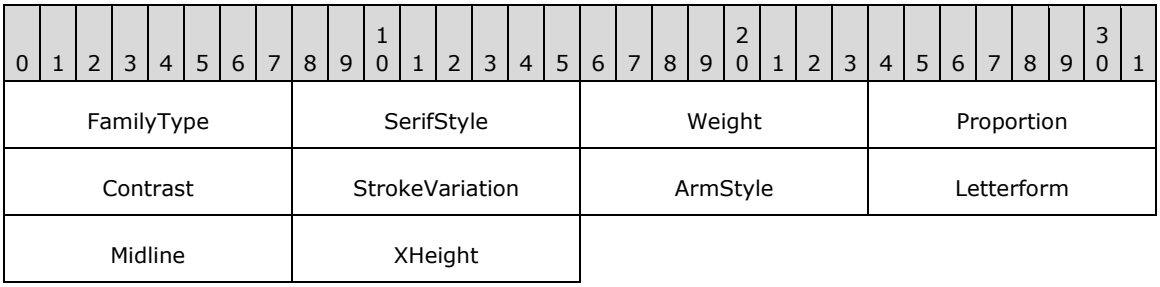
The LogPenEx object includes the specification of brush attributes, so it can be used to draw lines that consist of custom or predefined patterns. The following table shows the relationship between the **BrushStyle**, **ColorRef**, and **BrushHatch** fields in this object. Only supported brush styles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

BrushStyle	ColorRef	BrushHatch
BS_SOLID	A ColorRef object that specifies the color of lines drawn by the pen.	Not used and is ignored.
BS_NULL	Not used and is ignored.	Not used and is ignored.
BS_HATCHED	A ColorRef object that specifies the foreground color of the hatch pattern.	A value from the HatchStyle enumeration (section 2.1.17) that specifies the orientation of lines used to create the hatch. If **PS_GEOMETRIC** is not set in the **PenStyle** field, this field MUST be either **HS_SOLIDTEXTCLR (0x0008)** or

BrushStyle	ColorRef	BrushHatch
		HS_SOLIDBKCLR (0x000A).
BS_PATTERN	The low-order 16-bits is a value from the ColorUsage enumeration ([MS-WMF] section 2.1.1.6).	Not used and is ignored.
BS_DIBPATTERN	The low-order 16 bits is a value from the ColorUsage enumeration.	Not used and is ignored.
BS_DIBPATTERNPT	The low-order word is be a value from the ColorUsage enumeration.	Not used and is ignored.

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.



FamilyType (1 byte): An unsigned integer that specifies the family type. This value is in the FamilyType (section 2.1.12) enumeration table.

SerifStyle (1 byte): An unsigned integer that specifies the serif style. This value is in the SerifType (section 2.1.30) enumeration table.

Weight (1 byte): An unsigned integer that specifies the weight of the font. This value is in the Weight (section 2.1.34) enumeration table.

Proportion (1 byte): An unsigned integer that specifies the proportion of the font. This value is in the Proportion (section 2.1.28) enumeration table.

Contrast (1 byte): An unsigned integer that specifies the contrast of the font. This value is in the Contrast (section 2.1.8) enumeration table.

StrokeVariation (1 byte): An unsigned integer that specifies the stroke variation for the font. This value is in the StrokeVariation (section 2.1.33) enumeration table.

ArmStyle (1 byte): An unsigned integer that specifies the arm style of the font. This value is in the ArmStyle (section 2.1.3) enumeration table.

Letterform (1 byte): An unsigned integer that specifies the letterform of the font. This value is in the Letterform (section 2.1.20) enumeration table.

Midline (1 byte): An unsigned integer that specifies the midline of the font. This value is in the MidLine (section 2.1.23) enumeration table.

XHeight (1 byte): An unsigned integer that specifies the x height of the font. This value is 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
nSize																nVersion															
dwFlags																															
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): An unsigned integer that specifies the size in bytes, of this data structure.

nVersion (2 bytes): An unsigned 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
P	F	SO	G	M	W	S	D	C	DA	DS	A	SL	SC	SE	SP	0	0	0	0	0	0	0	0	0	0	0	0	SD	DD	DP	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.<48>

Value	Description
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. <49> The **PFD_SUPPORT_GDI** flag and **PFD_DOUBLEBUFFER** MUST NOT both be set.
SO PFD_SUPPORT_OPENGL	The pixel buffer supports OpenGL [OPENGL] drawing.
F PFD_GENERIC_FORMAT	The pixel format is natively supported by the operating system; this is known as the "generic" implementation.<50> 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 PFD_SWAP_EXCHANGE	The contents of the back buffer have been exchanged with the 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 accellerated 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.<51>
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.

Value	Description
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.<52>

cAlphaShift (1 byte): Specifies the shift count for alpha bitplanes in each RGBA color buffer.<53>

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.<54>

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.

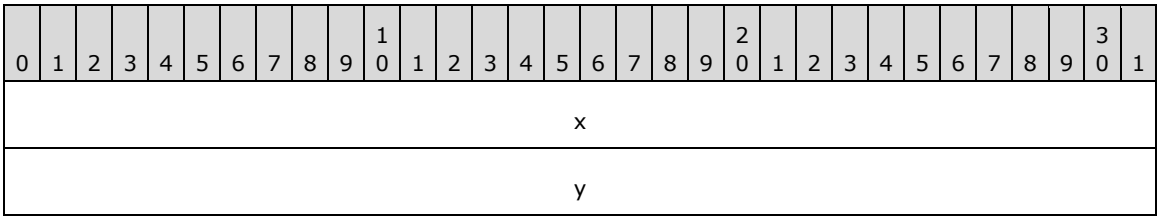
dwVisibleMask (4 bytes): Specifies the transparent color or index of an underlay plane. When the pixel type is RGBA, **dwVisibleMask** is a transparent RGB color value. When the pixel type is color index, it is a transparent index value.

dwDamageMask (4 bytes): This field SHOULD be ignored.

The PixelFormatDescriptor object is used in EMR_HEADER records (section 2.3.4.2) to specify the pixel format of the output surface.

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.

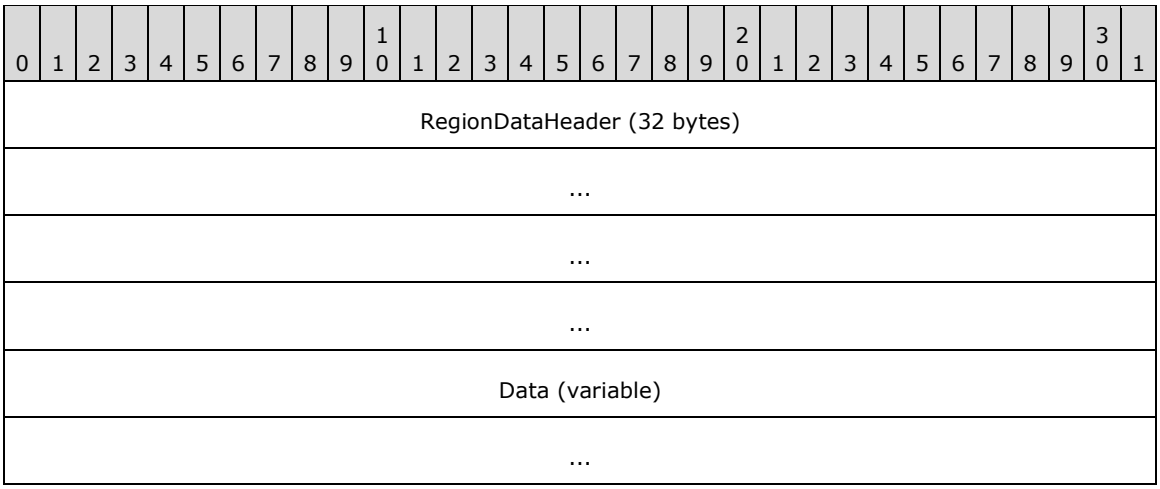


x (4 bytes): A BitFIX28_4 object (section 2.2.1) that represents the horizontal coordinate of the point.

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.



RegionDataHeader (32 bytes): A 256-bit RegionDataHeader object (section 2.2.25) that defines the contents of the **Data** field.

Data (variable): An array of 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 defines the properties of a RegionData (section 2.2.24) object.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Type																															
CountRects																															
RgnSize																															
Bounds																															
...																															
...																															
...																															

Size (4 bytes): An unsigned integer that specifies the size of this object in bytes. This value is 0x00000020.

Type (4 bytes): An unsigned integer that specifies the region type. This value is 0x00000001.

CountRects (4 bytes): An unsigned integer that specifies the number of rectangles in this region.

RgnSize (4 bytes): An unsigned integer that specifies the size of the buffer of rectangles in bytes.

Bounds (16 bytes): A 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
x																															
y																															
Red																Green															
Blue																Alpha															

x (4 bytes): A signed integer that specifies the horizontal position, in logical units.

y (4 bytes): A signed integer that specifies the vertical position, in logical units.

Red (2 bytes): An unsigned integer that specifies the red color value for the point.

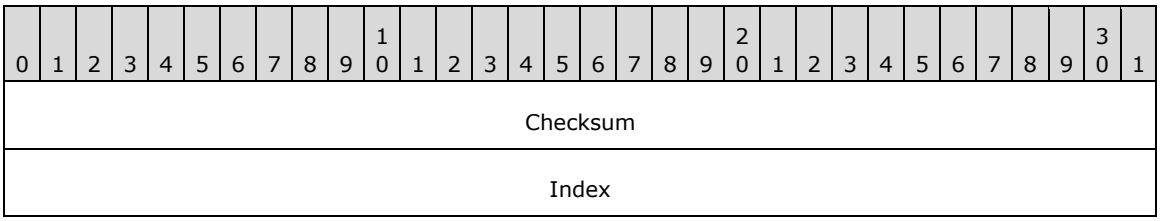
Green (2 bytes): An unsigned integer that specifies the green color value for the point.

Blue (2 bytes): An unsigned integer that specifies the blue color value for the point.

Alpha (2 bytes): An 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): An 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 ≤ *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.<55>

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

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): An 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
M11																															
M12																															
M21																															
M22																															
Dx																															
Dy																															

M11 (4 bytes): A FLOAT matrix value.

M12 (4 bytes): A FLOAT matrix value.

M21 (4 bytes): A FLOAT matrix value.

M22 (4 bytes): A FLOAT matrix value.

Dx (4 bytes): A FLOAT value that contains a horizontal translation component, in logical units.

Dy (4 bytes): A FLOAT 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'):

```
X' = M11 * X + M21 * Y + Dx
Y' = M12 * X + M22 * Y + Dy
```

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

2.3 EMF Records

This section specifies the 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.

Name	Section	Description
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 [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.

All EMF records MUST be multiples of 4 bytes in length; hence, each record starts on a 32-bit offset from the start of the metafile. To ensure each subsequent record also starts on a 32-bit boundary, an **AlignmentPadding** field is used, if necessary. The contents of **AlignmentPadding** fields are indeterminate and MUST be ignored. In general, such fields are shown only in the generic definitions of record categories.

2.3.1 Bitmap Record Types

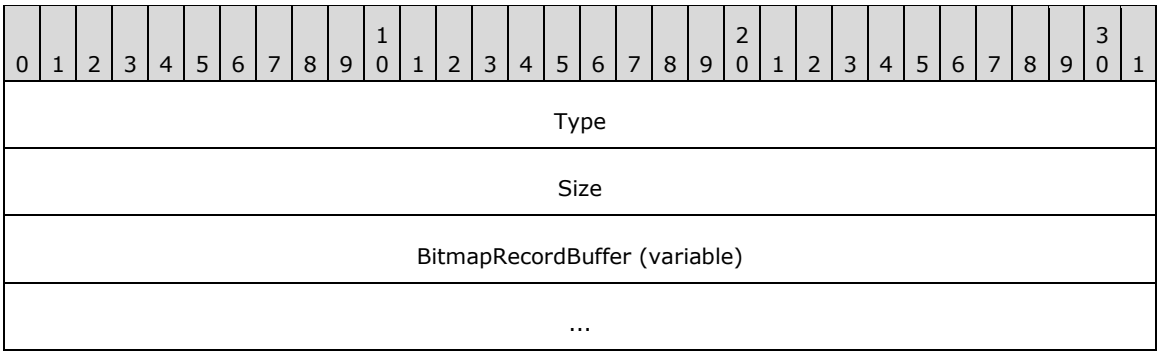
The **Bitmap** record types perform block transfers of bitmap images.

The following are the 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.
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_SETDIBITSTODEVICE	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

Name	Section	Description
		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): An 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
EMR_MASKBLT	0x0000004E
EMR_PLGBLT	0x0000004F
EMR_SETDIBITSTODEVICE	0x00000050
EMR_STRETCHDIBITS	0x00000051
EMR_ALPHABLEND	0x00000072
EMR_TRANSPARENTBLT	0x00000074

Size (4 bytes): An 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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BitmapRecordParm (variable)																															
...																															
AlignmentPadding (variable, optional)																															
...																															

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

AlignmentPadding (variable, optional): An 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 clipping regions used by bitmap records are maintained in a **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).
- If the color format of the source or pattern bitmap does not match the color format of the destination, the source pr 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 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.

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

See section 2.3 for more 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.<56>

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															

Bounds
...
...
...
xDest
yDest
cxDest
cyDest
BLENDFUNCTION
xSrc
ySrc
XformSrc (24 bytes)
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ALPHABLEND**. This value is 0x00000072.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

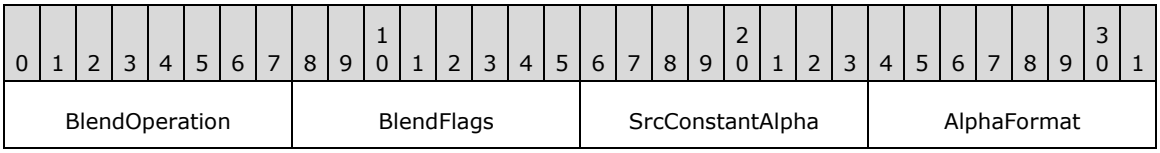
xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): A signed integer that specifies the logical width of the destination rectangle. This value MUST be greater than zero.

cyDest (4 bytes): A 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.



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 is 0x00 and MUST be ignored.

SrcConstantAlpha (1 byte): An 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 signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes of the source bitmap header.

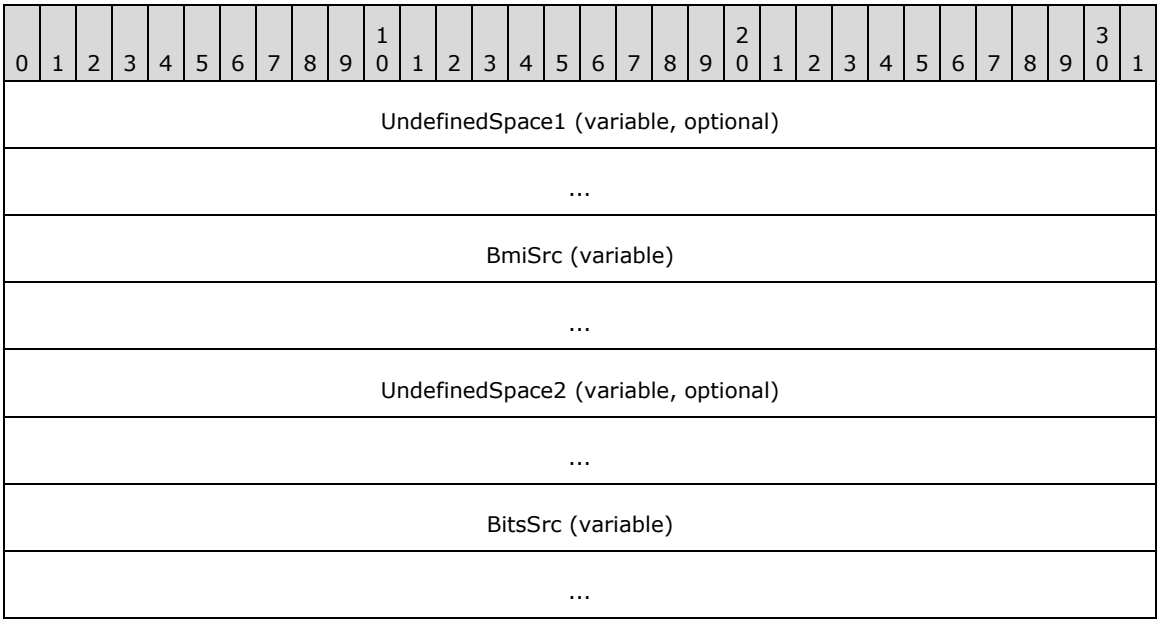
offBitsSrc (4 bytes): An unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits in the **BitmapBuffer** field.

cbBitsSrc (4 bytes): An unsigned integer that specifies the size in bytes of the source bitmap bits.

cxSrc (4 bytes): A signed integer that specifies the logical width of the source rectangle. This value MUST be greater than zero.

cySrc (4 bytes): A 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.



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
```

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

See section 2.3.1 for more 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.

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
cxDest																															
cyDest																															
BitBltRasterOperation																															
xSrc																															
ySrc																															
XformSrc (24 bytes)																															
...																															
...																															
...																															
BkColorSrc																															
UsageSrc																															
offBmiSrc																															
cbBmiSrc																															

offBitsSrc
cbBitsSrc
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_BITBLT. This value is 0x0000004C.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): A signed integer that specifies the logical width of the source and destination rectangles.

cyDest (4 bytes): A signed integer that specifies the logical height of the source and destination rectangles.

BitBltRasterOperation (4 bytes): An 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 is in the Ternary Raster Operation enumeration ([MS-WMF] section 2.1.1.31).

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

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap header in the **BitmapBuffer** field.

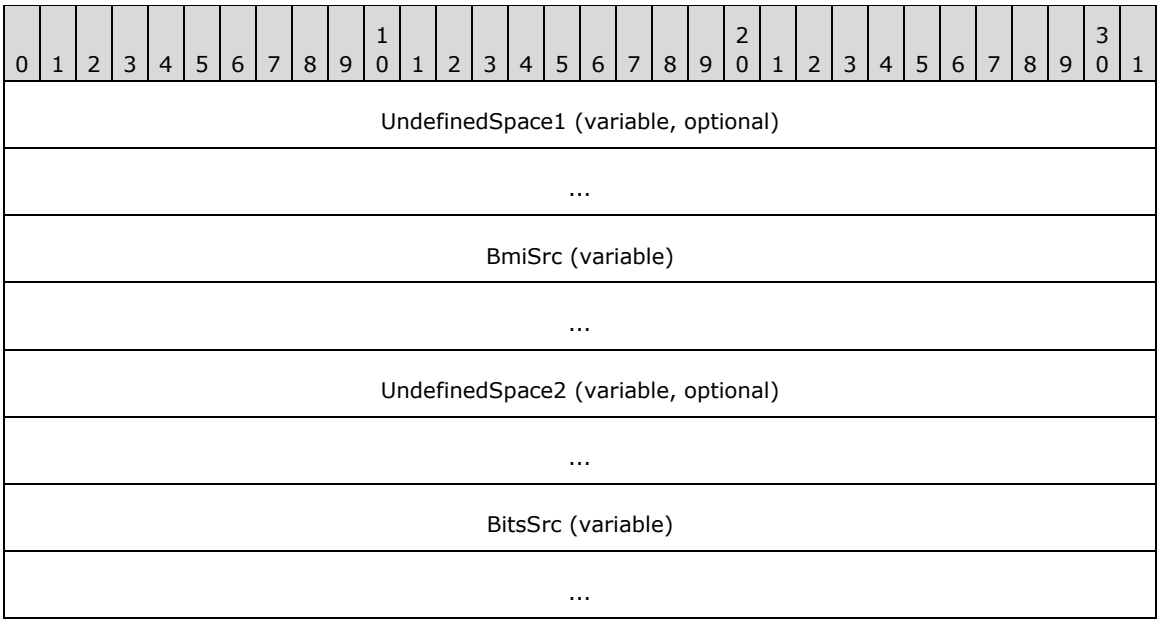
cbBmiSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An 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): An 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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.

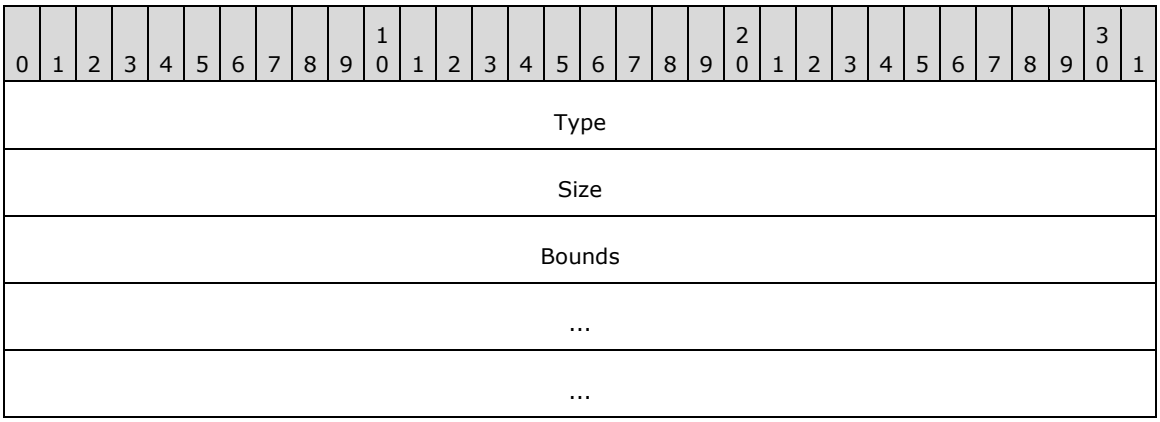
BitsSrc (variable): The source bitmap bits.

See section 2.3.1 for more 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.

Fields not specified in this section are specified in section 2.3.1.



...
xDest
yDest
cxDest
cyDest
ROP4
xSrc
ySrc
XformSrc (24 bytes)
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask

BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_MASKBLT**. This value is 0x0000004E.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): A signed integer that specifies the logical width of the destination rectangle.

cyDest (4 bytes): A 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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved																BackgroundROP3								ForegroundROP3							

Reserved (2 bytes): This field SHOULD be 0x0000 and MUST be ignored.<57>

BackgroundROP3 (1 byte): The unsigned, most-significant 8 bits of a 24-bit ternary raster operation value from the 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 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 signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes of the source bitmap header.

offBitsSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the source bitmap bits.

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

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

UsageMask (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the mask bitmap header. This value is in the DIBColors enumeration.

offBmiMask (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the mask bitmap header.

offBitsMask (4 bytes): An 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): An 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. Thus, 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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															
BmiSrc (variable)																															
...																															
UndefinedSpace2 (variable, optional)																															
...																															
BitsSrc (variable)																															
...																															
UndefinedSpace3 (variable, optional)																															

...
BmiMask (variable)
...
UndefinedSpace4 (variable, optional)
...
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 more 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.

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
aptlDest (24 bytes)																															

...
...
...
xSrc
ySrc
cxSrc
cySrc
XformSrc (24 bytes)
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask
BitmapBuffer (variable)

...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_PLGBLT**. This value is 0x0000004F.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

aptlDest (24 bytes): An array of three 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 signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

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

cxSrc (4 bytes): A signed integer that specifies the logical width of the source rectangle.

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the source bitmap.

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

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

UsageMask (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the mask bitmap header. This value is in the DIBColors enumeration.

offBmiMask (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the mask bitmap header.

offBitsMask (4 bytes): An 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): An 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_PLGBLT record or with each other. Thus, 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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															
BmiSrc (variable)																															
...																															
UndefinedSpace2 (variable, optional)																															
...																															
BitsSrc (variable)																															
...																															
UndefinedSpace3 (variable, optional)																															
...																															
BmiMask (variable)																															
...																															
UndefinedSpace4 (variable, optional)																															
...																															
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 more 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.

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
xSrc																															
ySrc																															
cxSrc																															
cySrc																															
offBmiSrc																															
cbBmiSrc																															
offBitsSrc																															

cbBitsSrc
UsageSrc
iStartScan
cScans
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETDIBITSTODEVICE**. This value is 0x00000050.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

xSrc (4 bytes): A signed integer that specifies the x-coordinate in pixels of the lower-left corner of the source rectangle.

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

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

cySrc (4 bytes): A signed integer that specifies the height in pixels of the source rectangle.

offBmiSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An 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): An unsigned integer that specifies the size in bytes, of the source bitmap bits.

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

iStartScan (4 bytes): An unsigned integer that specifies the first scan line in the array.

cScans (4 bytes): An 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															
BmiSrc (variable)																															
...																															
UndefinedSpace2 (variable, optional)																															
...																															
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 for more 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.

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															

yDest
cxDest
cyDest
BitBltRasterOperation
xSrc
ySrc
XformSrc (24 bytes)
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_STRETCHBLT**. This value is 0x0000004D.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): A signed integer that specifies the logical width of the destination rectangle.

cyDest (4 bytes): A signed integer that specifies the logical height of the destination rectangle.

BitBltRasterOperation (4 bytes): An 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 is in the Ternary Raster Operation enumeration ([MS-WMF] section 2.1.1.31).

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

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap header.

cbBmiSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap bits.

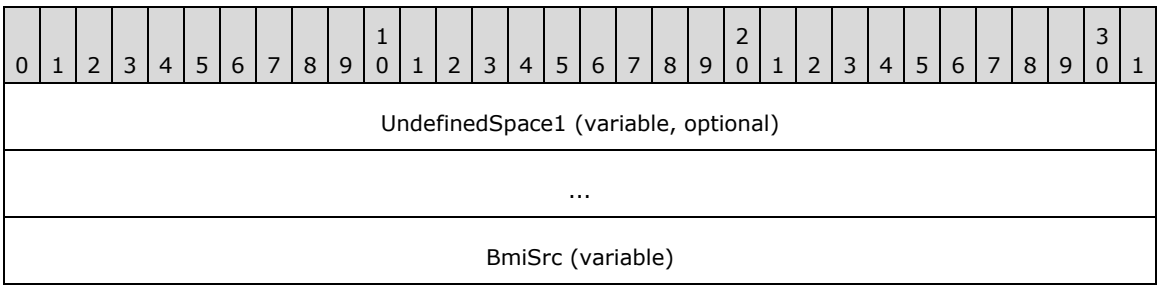
cbBitsSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap bits.

cxSrc (4 bytes): A signed integer that specifies the logical width of the source rectangle.

cySrc (4 bytes): A 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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



...
UndefinedSpace2 (variable, optional)
...
BitsSrc (variable)
...

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

See section 2.3.1 for more 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.

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
xSrc																															
ySrc																															
cxSrc																															
cySrc																															

offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
UsageSrc
BitBltRasterOperation
cxDest
cyDest
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_STRETCHDIBITS**. This value is 0x00000051.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

xSrc (4 bytes): A signed integer that specifies the x-coordinate in pixels of the upper-left corner of the source rectangle.

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

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

cySrc (4 bytes): A signed integer that specifies the height in pixels of the source rectangle.

offBmiSrc (4 bytes): An unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

cbBmiSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap bits.

cbBitsSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap bits.

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

BitBltRasterOperation (4 bytes): An 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.

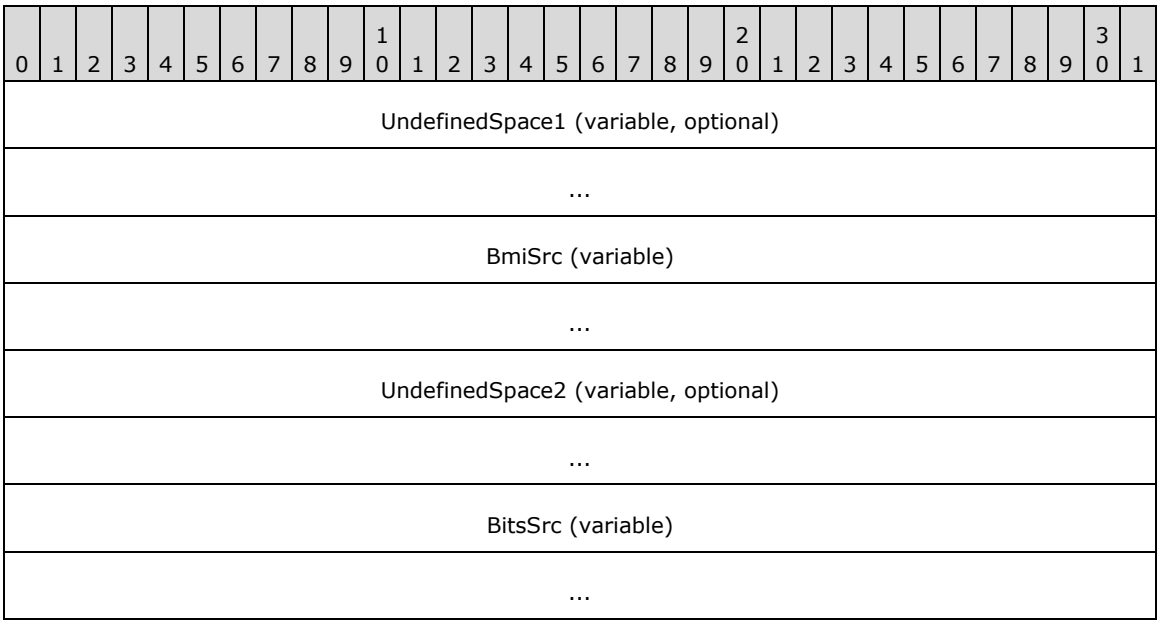
This value is in the Ternary Raster Operation enumeration ([MS-WMF] section 2.1.1.31).

cxDest (4 bytes): A signed integer that specifies the logical width of the destination rectangle.

cyDest (4 bytes): A 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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.

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 more 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.<58>

Fields not specified in this section are specified in section 2.3.1.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
cxDest																															
cyDest																															
TransparentColor																															
xSrc																															
ySrc																															
XformSrc (24 bytes)																															
...																															
...																															
...																															
BkColorSrc																															
UsageSrc																															
offBmiSrc																															

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

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_TRANSPARENTBLT**. This value is 0x00000074.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping regions (section 3.1.1.2.1) in the playback device context (section 3.1) is empty, this record has no effect.

xDest (4 bytes): A signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): A signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): A signed integer that specifies the logical width of the destination rectangle.

cyDest (4 bytes): A signed integer that specifies the logical height of the destination rectangle.

TransparentColor (4 bytes): A ColorRef object ([MS-WMF] section 2.2.2.8) that specifies the color in the source bitmap to be treated as transparent.

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

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

UsageSrc (4 bytes): An unsigned integer that specifies how to interpret values in the color table in the source bitmap header. This value is in the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap header.

cbBmiSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap header.

offBitsSrc (4 bytes): An unsigned integer that specifies the offset in bytes, from the start of this record to the source bitmap bits.

cbBitsSrc (4 bytes): An unsigned integer that specifies the size in bytes, of the source bitmap bits.

cxSrc (4 bytes): A signed integer that specifies the logical width of the source rectangle.

cySrc (4 bytes): A 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. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

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.<59>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															
BmiSrc (variable)																															
...																															
UndefinedSpace2 (variable, optional)																															
...																															
BitsSrc (variable)																															
...																															

BmiSrc (variable): The source bitmap header.

BitsSrc (variable): The source bitmap bits.

See section 2.3.1 for more bitmap record types.

2.3.2 Clipping Record Types

The **Clipping** record types define and manage clipping regions. The clipping regions used by clipping record types are part of the **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).

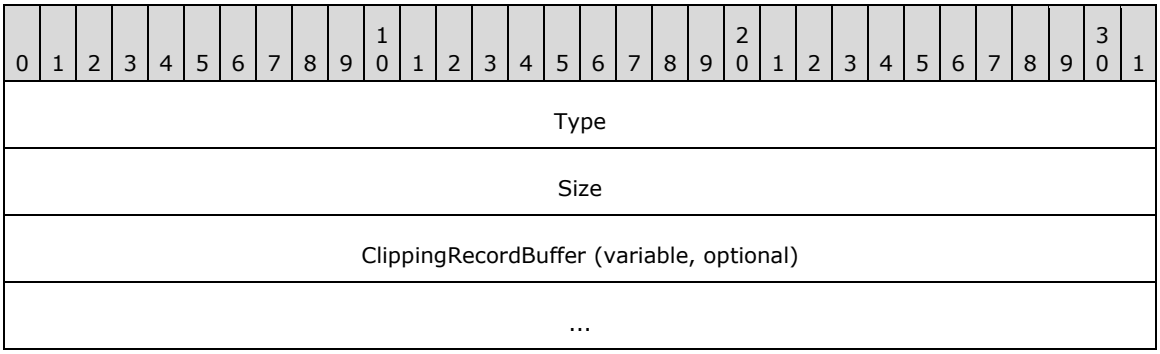
Note: The EMR_SETMETARGN record does not specify parameters.

The following are the clipping record types.

Name	Section	Description
EMR_EXCLUDECLIPRECT	2.3.2.1	Specifies a clipping region that consists of the current clipping region minus the specified rectangle.
EMR_EXTSELECTCLIPRGN	2.3.2.2	Combines the specified region with the current clipping region, using the specified mode.

Name	Section	Description
EMR_INTERSECTCLIPRECT	2.3.2.3	Specifies a clipping region from the intersection of the current clipping region and the specified rectangle.
EMR_OFFSETCLIPRGN	2.3.2.4	Specifies a clipping region as the current clipping region moved by a specified offset.
EMR_SELECTCLIPPATH	2.3.2.5	Specifies a clipping region as the current clipping region combined with the current path bracket, using the specified mode.
EMR_SETMETARGN	2.3.2	If the current metaregion is null, it is set to the current clipping region. Otherwise, the current metaregion is intersected with the current clipping region, and the result is the new metaregion. After the operation, the current clipping region is set to null. During playback, drawing occurs only within the intersection of the metaregion and clipping region. This EMF record specifies no parameters.

The generic structure of clipping records is specified as follows.



Type (4 bytes): An 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
EMR_INTERSECTCLIPRECT	0x0000001E
EMR_SELECTCLIPPATH	0x00000043
EMR_EXTSELECTCLIPRGN	0x0000004B

Size (4 bytes): An 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, optional): An array of bytes that contains the data for the clipping record. The size of this field is a multiple of 4 bytes.

The EMR_SETMETARGN record does not contain this field.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ClippingRecordParm (variable)																															
...																															
AlignmentPadding (variable, optional)																															
...																															

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

AlignmentPadding (variable, optional): An 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 more EMF record types.

2.3.2.1 EMR_EXCLUDECLIPRECT Record

The **EMR_EXCLUDECLIPRECT** record excludes the specified rectangle from the current clipping region.

Fields not specified in this section are specified in section 2.3.2.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Clip																															
...																															
...																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_EXCLUDECLIPRECT. This value is 0x0000001D.

Clip (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies a rectangle in logical units.

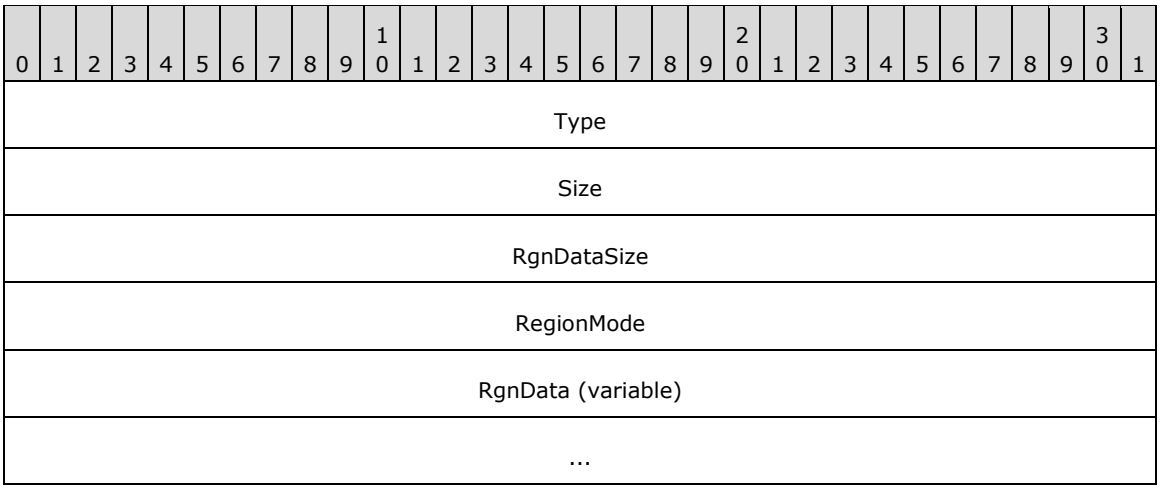
The result of the intersection is saved as the new current clipping region. The lower and right edges of the specified rectangle MUST NOT be excluded from clipping.

See section 2.3.2 for more clipping record types.

2.3.2.2 EMR_EXTSELECTCLIPRGN Record

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

Fields not specified in this section are specified in section 2.3.2.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_EXTSELECTCLIPRGN. This value is 0x0000004B.

RgnDataSize (4 bytes): An unsigned integer that specifies the size of the **RgnData** field in bytes.

RegionMode (4 bytes): An unsigned integer that specifies the way to use the region. This value is in the RegionMode (section 2.1.29) enumeration.

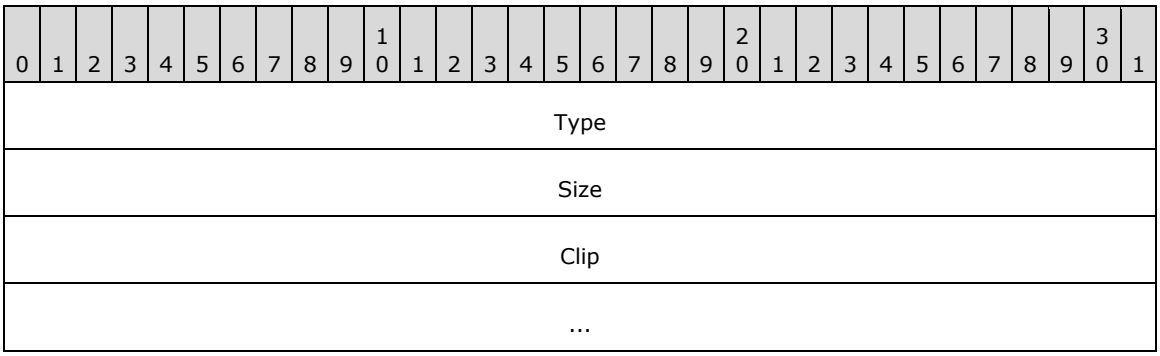
RgnData (variable): An array of bytes that specifies a **RegionData** object (section 2.2.24) in logical units. If **RegionMode** is **RGN_COPY**, this data can be omitted and the clipping region SHOULD be set to the default clipping region.

See section 2.3.2 for more 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.

Fields not specified in this section are specified in section 2.3.2.



...
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_INTERSECTCLIPRECT**. This value is 0x0000001E.

Clip (16 bytes): A 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 more 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.

Fields not specified in this section are specified in section 2.3.2.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Offset																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_OFFSETCLIPRGN. This value is 0x0000001A.

Offset (8 bytes): A 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 more clipping record types.

2.3.2.5 EMR_SELECTCLIPPATH Record

The **EMR_SELECTCLIPPATH** record sets the current clipping region in the playback device context to the current clipping region combined with current path bracket.

Fields not specified in this section are specified in section 2.3.2.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															

RegionMode

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SELECTCLIPPATH. This value is 0x00000043.

RegionMode (4 bytes): An unsigned integer that specifies how to combine the current clipping region with the current path bracket. This value is in the RegionMode enumeration (section 2.1.29).

See section 2.3.2 for more 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 ([MS-EMFPLUS] section 2.3).
EMR_COMMENT_EMFSPOOL	2.3.3.3	Contains embedded EMFSPOOL records ([MS-EMFSPOOL] section 2.2).
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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
CommentRecordBuffer (variable)																															
...																															

Type (4 bytes): An unsigned integer from the RecordType enumeration (section 2.1.1) that identifies this record as a comment record. This value is 0x00000046.

Size (4 bytes): An 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): An 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.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
CommentIdentifier (optional)																															
CommentRecordParm (variable)																															
...																															
AlignmentPadding (variable, optional)																															
...																															

CommentIdentifier (4 bytes, optional): An 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 is processed as an EMR_COMMENT record.

Name	Value
EMR_COMMENT_EMFSPOOL	0x00000000
EMR_COMMENT_EMFPLUS	0x2B464D45
EMR_COMMENT_PUBLIC	0x43494447

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

AlignmentPadding (variable, optional): An 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 more EMF record types.

2.3.3.1 EMR_COMMENT Record

The **EMR_COMMENT** record contains arbitrary private data.

Fields not specified in this section are specified in section 2.3.3.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
PrivateData (variable, optional)																															
...																															

PrivateData (variable, optional): An array of bytes that specifies the private data. The first 32-bit field of this data MUST NOT be one of the predefined comment identifier values specified in section 2.3.3.

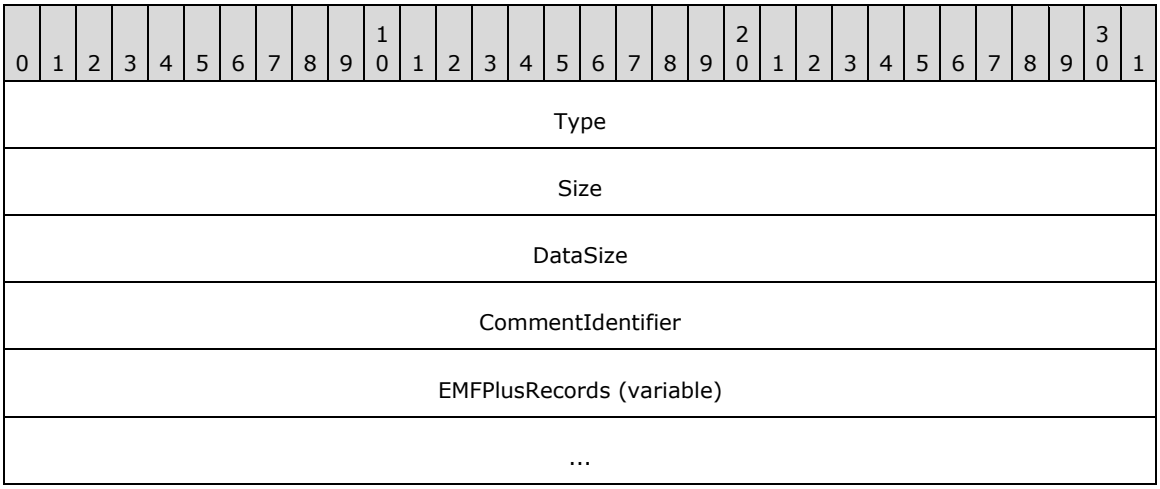
Private data is unknown to EMF; it is meaningful only to applications that know the format of the data and how to use it. EMR_COMMENT private data records MAY<60> be ignored.

See section 2.3.3 for more comment record types.

2.3.3.2 EMR_COMMENT_EMFPLUS Record

The **EMR_COMMENT_EMFPLUS** record contains embedded EMF+ records ([MS-EMFPLUS] section 2.3).

Fields not specified in this section are specified in section 2.3.3.



CommentIdentifier (4 bytes): An 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.

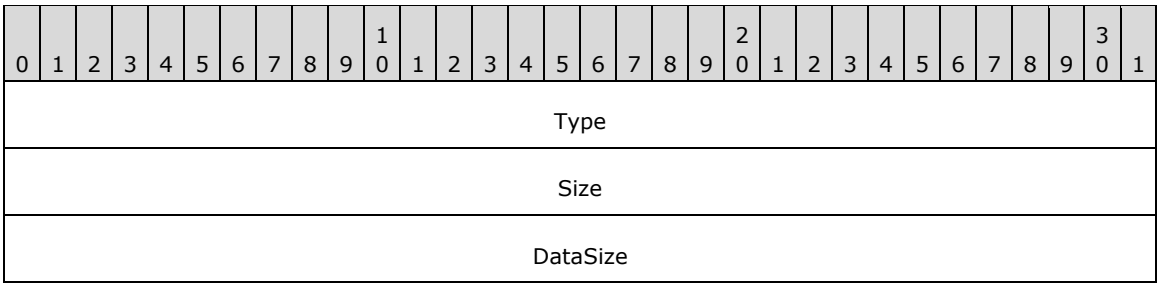
EMFPlusRecords (variable): An array of bytes that contains one or more EMF+ records.

See section 2.3.3 for more comment record types.

2.3.3.3 EMR_COMMENT_EMFSPOOL Record

The **EMR_COMMENT_EMFSPOOL** record contains embedded EMFSPOOL records ([MS-EMFSPOOL] section 2.2).

Fields not specified in this section are specified in section 2.3.3.



CommentIdentifier
EMFSpoolRecordIdentifier
EMFSpoolRecords (variable)
...

CommentIdentifier (4 bytes): An 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): An unsigned integer that identifies the type of EMR_COMMENT_EMFSPOOL record. The value 0x544F4E46, which is the ASCII string "TONE", identifies this as an EMFSPOOL font definition record ([MS-EMFSPOOL] section 2.2.3.3).

EMFSpoolRecords (variable): An array of bytes that contain one or more font definition records.

See section 2.3.3 for more 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_ENDGROUP	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.

Fields not specified in this section are specified in section 2.3.3.

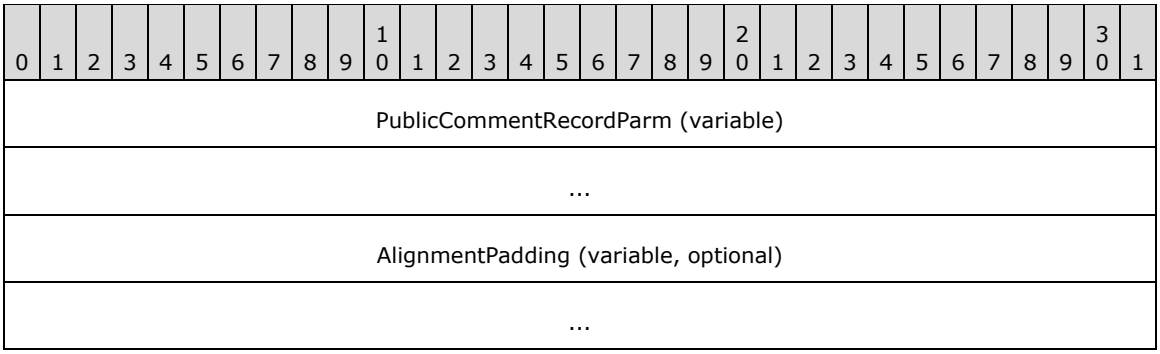
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
CommentIdentifier																															
PublicCommentIdentifier																															
PublicCommentRecordBuffer (variable, optional)																															

...

CommentIdentifier (4 bytes): An 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): An 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, optional): An array of bytes that contains the remainder of the public comment record.



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

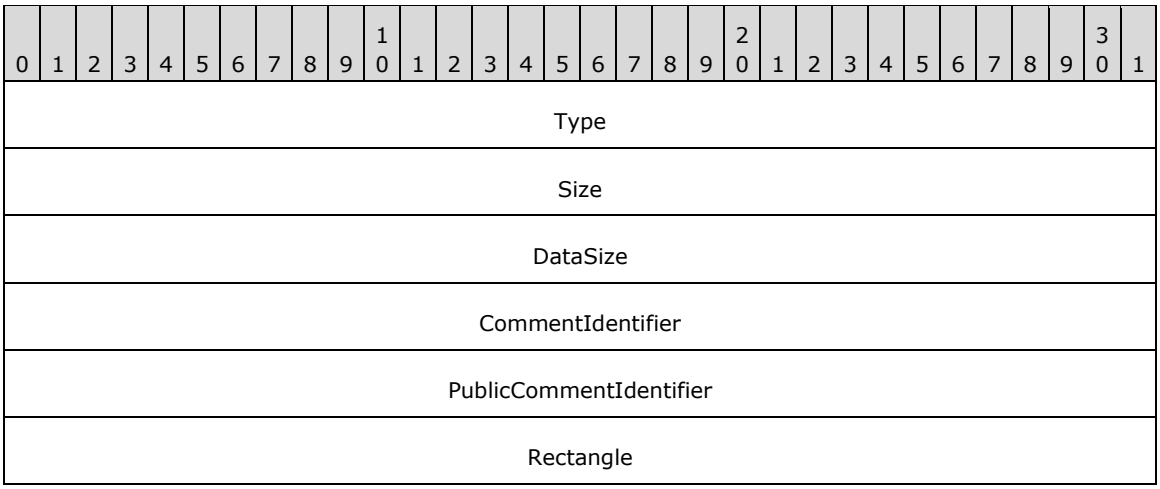
AlignmentPadding (variable, optional): An array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section 2.3.3 for more 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.

Fields not specified in this section are specified in section 2.3.3 or 2.3.3.4.



...
...
...
nDescription
Description (variable, optional)
...

PublicCommentIdentifier (4 bytes): An unsigned integer that identifies the type of public comment record as EMR_COMMENT_BEGINGROUP from the EmrComment enumeration (section 2.1.10). This value is 0x00000002.

Rectangle (16 bytes): A 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, optional): A null-terminated Unicode string that describes this group of records.

This record MUST be followed by a corresponding EMR_COMMENT_ENDGROUP record (section 2.3.3.4.2). These record groups can be nested.

See section 2.3.3.4 for more 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.

Fields not specified 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
CommentIdentifier																															
PublicCommentIdentifier																															

PublicCommentIdentifier (4 bytes): An unsigned integer that identifies the type of public comment record as EMR_COMMENT_ENDGROUP from the EmrComment enumeration (section 2.1.10). This value is 0x00000003.

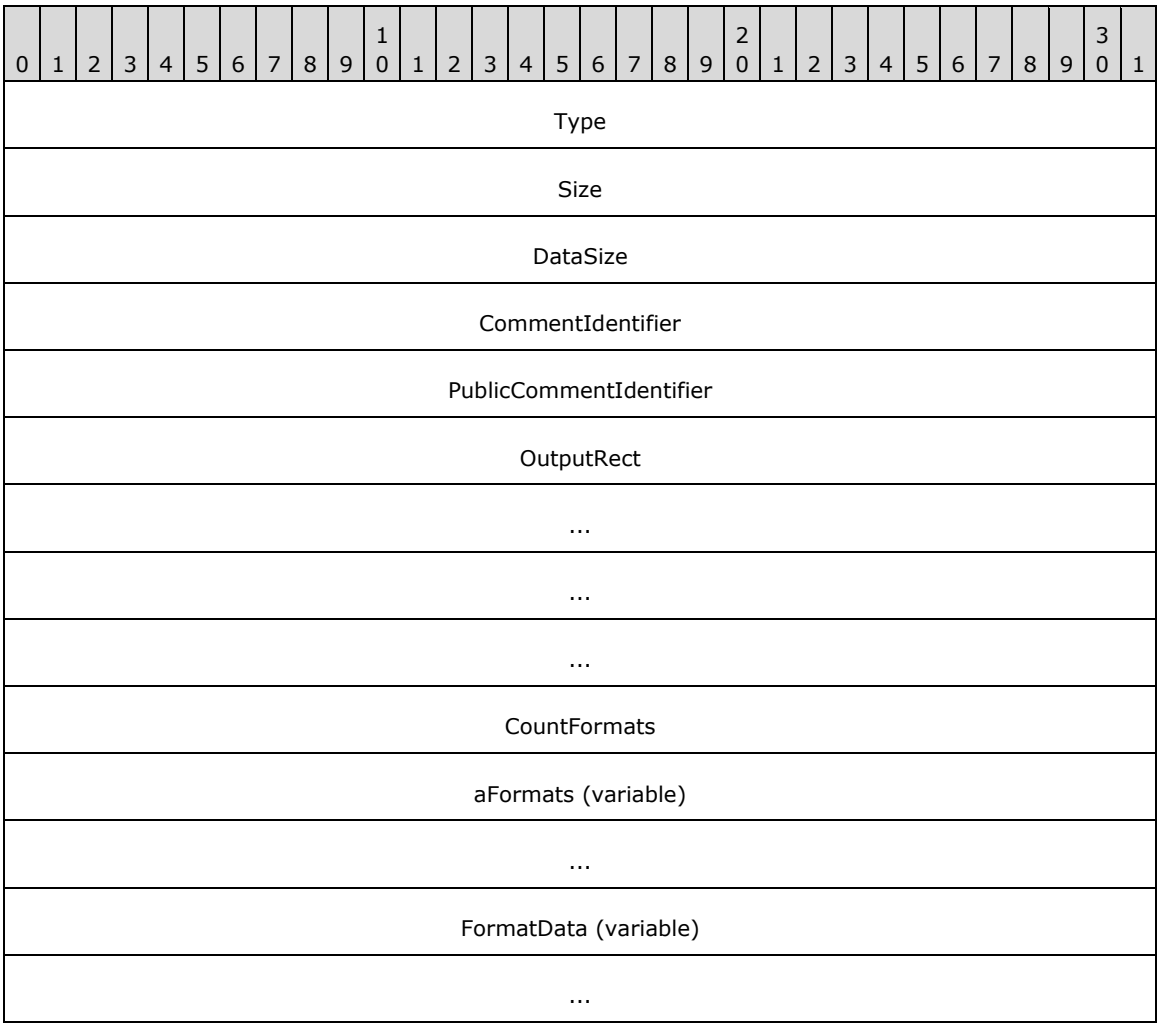
This record MUST be preceded by a corresponding. EMR_COMMENT_BEGINGROUP (section 2.3.3.4.1). These records can be nested.

See section 2.3.3.4 for more 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.

Fields not specified in this section are specified in section 2.3.3 or 2.3.3.4.



PublicCommentIdentifier (4 bytes): An unsigned integer that identifies the type of public comment record as EMR_COMMENT_MULTIFORMATS from the EmrComment enumeration (section 2.1.10). This value is 0x40000004.

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

CountFormats (4 bytes): An 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): The image data for all graphics formats contained in this record.

The size of the data for each image is specified by the **DataSize** field in the corresponding EmrFormat object. Thus, the total size of this field is the sum of **DataSize** 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.

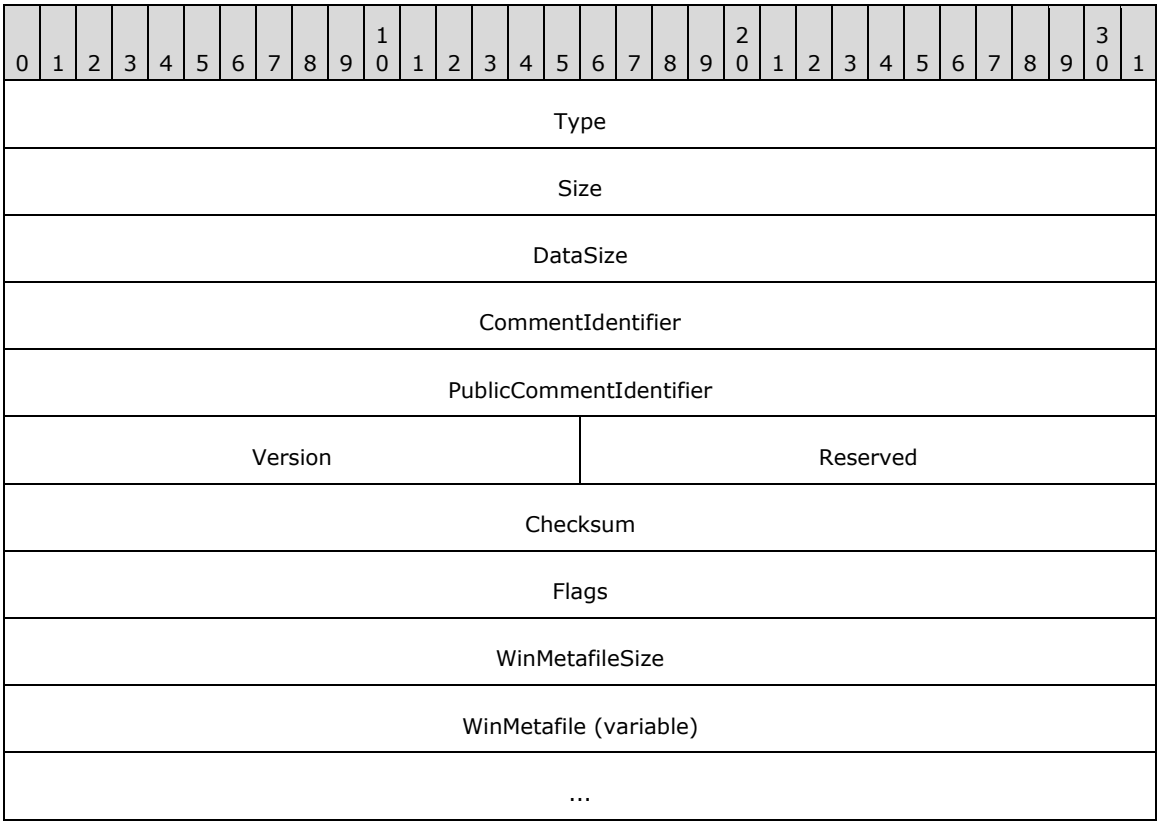
For example, an application can use this record type to specify an image in EPS format using EpsData objects (section 2.2.6). Subsequently, the PostScript version of the image can be rendered if that graphics format is supported by the printer driver on the playback system.<61>

See section 2.3.3.4 for more 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.

Fields not specified in this section are specified in section 2.3.3 or 2.3.3.4.



PublicCommentIdentifier (4 bytes): An unsigned integer that identifies the type of public comment record as EMR_COMMENT_WINDOWS_METAFILE from the EmrComment enumeration (section 2.1.10). This value is 0x80000001.

Version (2 bytes): An unsigned integer that specifies the WMF metafile version in terms of support for DIBs, from the MetafileVersion enumeration ([MS-WMF] section 2.1.1.19).

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

Checksum (4 bytes): An unsigned integer that specifies the checksum for this record.

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

WinMetafileSize (4 bytes): An unsigned integer that specifies the size in bytes, of the **WinMetafile** field.

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

See section 2.3.3.4 for more public comment record types.

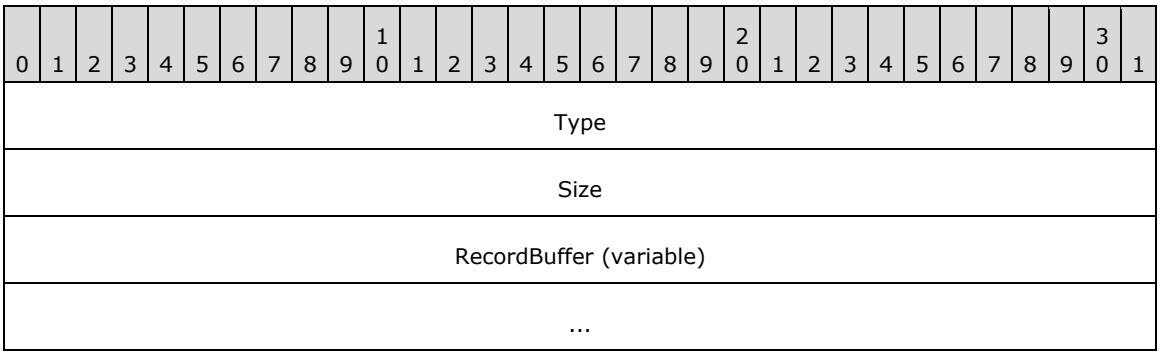
2.3.4 Control Record Types

The **Control** record types define the start and end of an EMF metafile and its properties.

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.

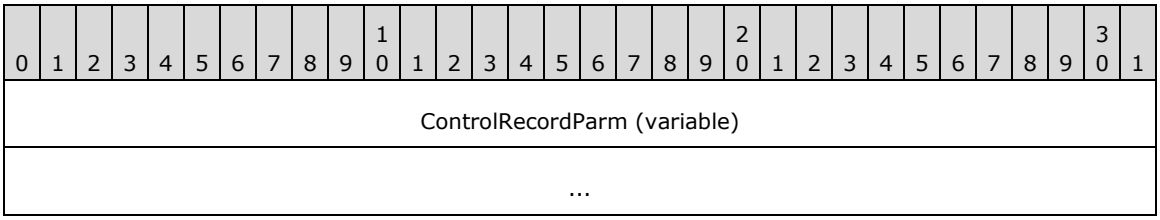


Type (4 bytes): An unsigned integer that defines the type of record. The control record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_HEADER	0x00000001
EMR_EOF	0x0000000E

Size (4 bytes): An 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.



AlignmentPadding (variable, optional)
...

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

AlignmentPadding (variable, optional): An 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 more EMF record types.

2.3.4.1 EMR_EOF Record

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

Fields not specified in this section are specified in section 2.3.4.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
nPalEntries																															
offPalEntries																															
PaletteBuffer (variable, optional)																															
...																															
SizeLast																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_EOF. This value is 0x0000000E.

nPalEntries (4 bytes): An unsigned integer that specifies the number of palette entries.

offPalEntries (4 bytes): An unsigned integer that specifies the offset to the palette entries from the start of this record.

PaletteBuffer (variable, optional): An array of bytes that contains palette data, which is not required to be contiguous with the fixed-length portion of the EMR_EOF record. Thus, fields in this buffer that are labeled "UndefinedSpace" are optional and MUST be ignored.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															

PaletteEntries (variable)
...
UndefinedSpace2 (variable, optional)
...

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

SizeLast (4 bytes): An 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 more control record types.

2.3.4.2 EMR_HEADER Record Types

The **EMR_HEADER** record is the starting point of an EMF metafile. It specifies properties of the device on which the image in the metafile was recorded; this information in the header record makes it possible for EMF metafiles to be independent of any specific output device.

The following are the EMR_HEADER record types.

Name	Section	Description
EmfMetafileHeader	2.3.4.2.1	The original EMF header record.
EmfMetafileHeaderExtension1	2.3.4.2.2	The header record defined in the first extension to EMF, which added support for OpenGL records and an optional internal pixel format descriptor.<62>
EmfMetafileHeaderExtension2	2.3.4.2.3	The header record defined in the second extension to EMF, which added the capability of measuring display dimensions in micrometers.<63>

EMF metafiles SHOULD be created with an **EmfMetafileHeaderExtension2** header record.

The generic structure of EMR_HEADER records is specified as follows.

Fields not specified in this section are specified in section 2.3.4.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
EmfHeader (80 bytes)																															
...																															
...																															

...
EmfHeaderRecordBuffer (variable, optional)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_HEADER**. This value is 0x00000001.

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

EmfHeaderRecordBuffer (variable, optional): An array of bytes that contains the remainder of the EMF header record.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
EmfHeaderRecordParm (variable)																															
...																															
AlignmentPadding (variable, optional)																															
...																															

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

AlignmentPadding (variable, optional): An 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 EmfMetafileHeader record. The fixed-size part of this header is 88 bytes, and it contains a Header object (section 2.2.9).
- 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 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).

There are one or two optional, variable-length fields that are possible in each 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.

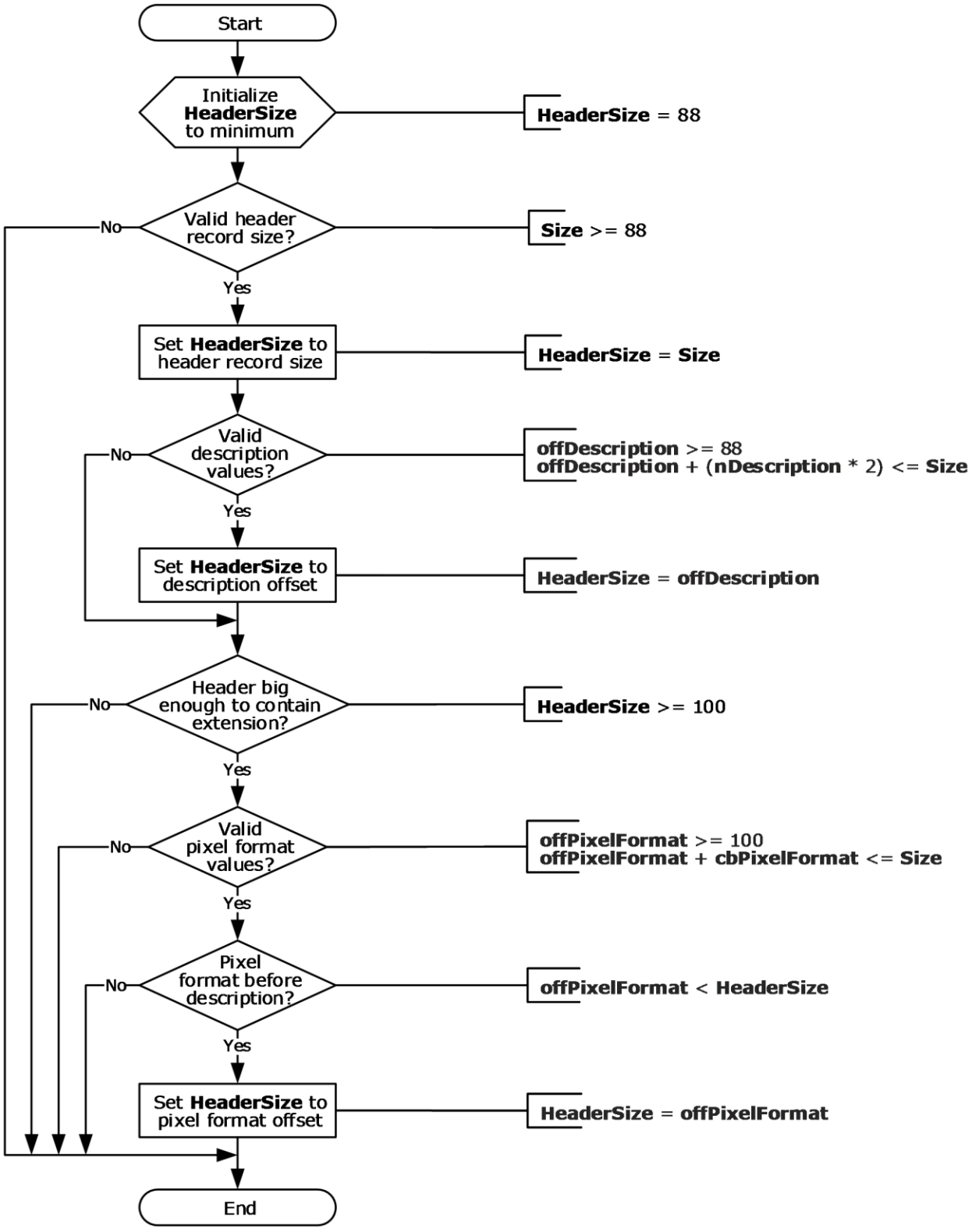


Figure 3: Header type determination algorithm

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

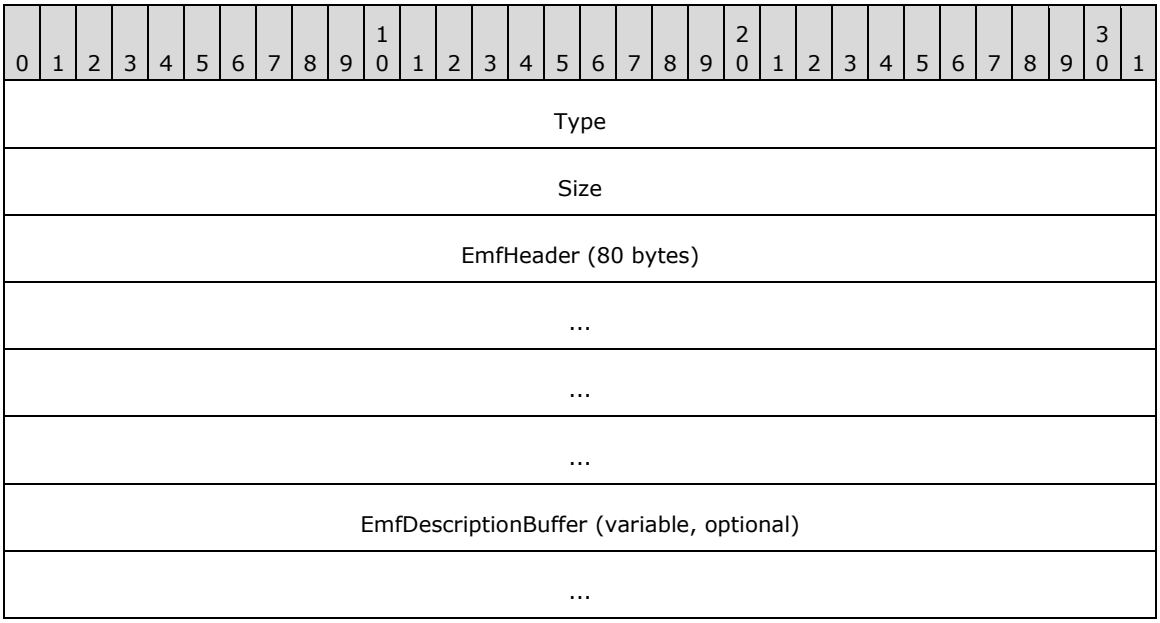
- If **HeaderSize** >= 108, the record type is EmfMetafileHeaderExtension2.
- If **HeaderSize** >= 100, the record type is EmfMetafileHeaderExtension1.
- Otherwise, the record type is EmfMetafileHeader.

See section 2.3.4 for more control record types.

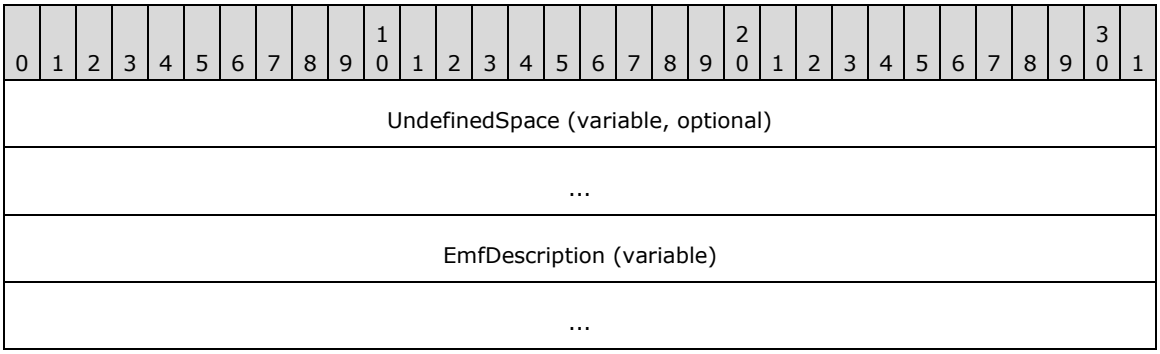
2.3.4.2.1 EmfMetafileHeader Record

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

Fields not specified in this section are specified in section 2.3.4 or 2.3.4.2.



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



EmfDescription (variable): A 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.

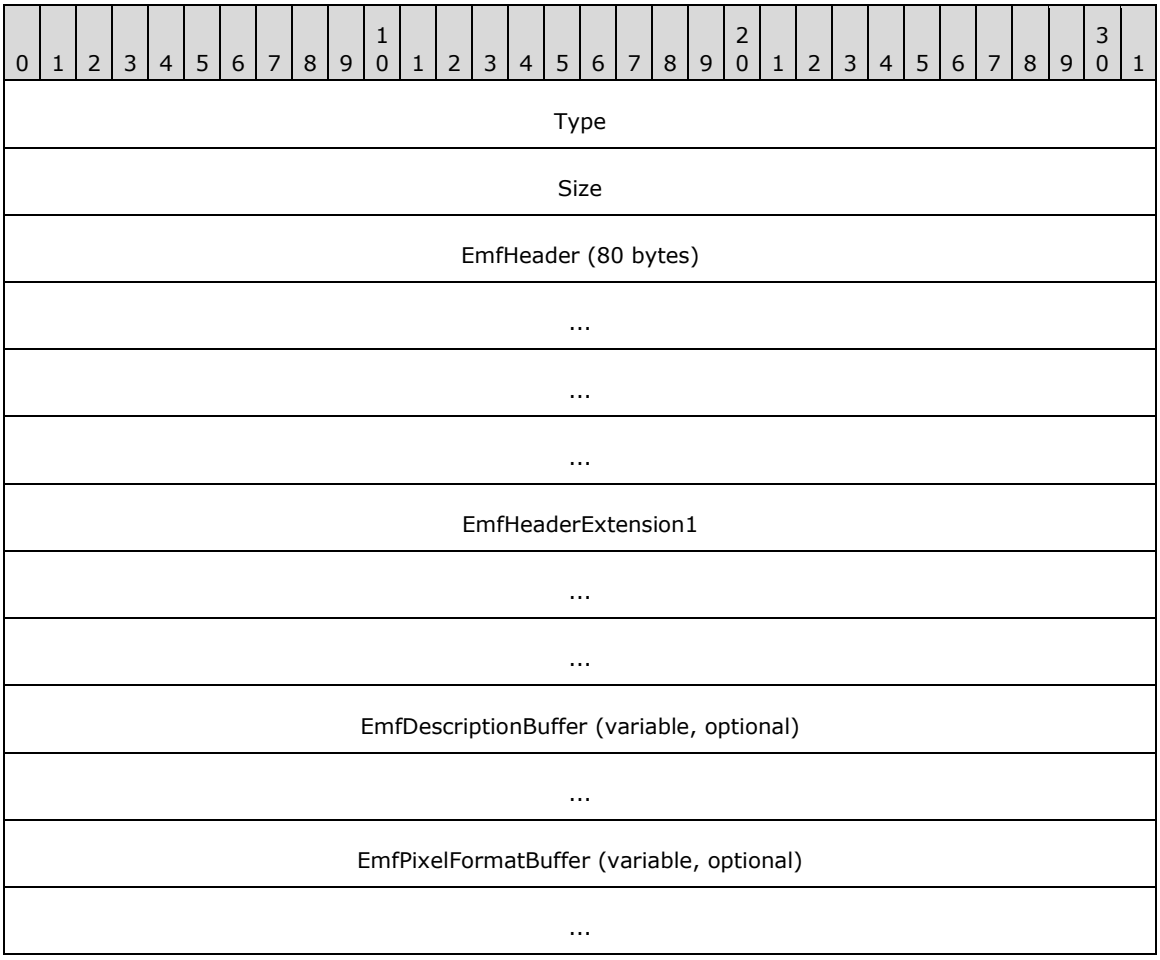
The value of the **Size** field can be used to distinguish between the different EMR_HEADER record types. See the flowchart in section 2.3.4.2 for details.

See section 2.3.4.2 for more header record types.

2.3.4.2.2 EmfMetafileHeaderExtension1 Record

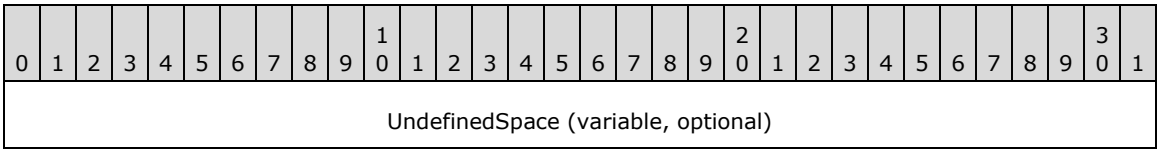
The **EmfMetafileHeaderExtension1** record is the header record used in the first extension to EMF metafiles. Following the **EmfHeaderExtension1** field, the remaining fields are optional and can be present in any order.

Fields not specified in this section are specified in section 2.3.4 or 2.3.4.2.



EmfHeaderExtension1 (12 bytes): A HeaderExtension1 object (section 2.2.10), which specifies additional information about the image in the metafile.

EmfDescriptionBuffer (variable, optional): An array of bytes that contains the EMF description string, which is not required to be contiguous with the fixed-length part of this record. Thus, the undefined space field in this buffer is optional and MUST be ignored.



...
EmfDescription (variable)
...

EmfDescription (variable): A 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, optional): An array of bytes that contains the EMF pixel format descriptor. It is not required to be contiguous with the **EmfHeaderExtension1** or **EmfDescription** fields. Thus, the undefined space field in this buffer is optional and MUST be ignored.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace2 (variable, optional)																															
...																															
EmfPixelFormat (40 bytes)																															
...																															
...																															
...																															

EmfPixelFormat (40 bytes): A PixelFormatDescriptor object (section 2.2.22), which specifies the 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 present in each 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 the flowchart in section 2.3.4.2 for details.

2.3.4.2.3 EmfMetafileHeaderExtension2 Record

The **EmfMetafileHeaderExtension2** record is the header record used in the second extension to EMF metafiles. Following the **EmfHeaderExtension2** field, the remaining fields are optional and can be present in any order.

Fields not specified 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
EmfHeader (80 bytes)																															
...																															
...																															
...																															
EmfHeaderExtension1																															
...																															
...																															
EmfHeaderExtension2																															
...																															
EmfDescriptionBuffer (variable, optional)																															
...																															
EmfPixelFormatBuffer (variable, optional)																															
...																															

EmfHeaderExtension1 (12 bytes): A HeaderExtension1 object (section 2.2.10), which specifies additional information about the image in the metafile.

EmfHeaderExtension2 (8 bytes): A HeaderExtension2 object (section 2.2.11), which specifies additional information about the image in the metafile.

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

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace1 (variable, optional)																															
...																															
EmfDescription (variable)																															

...

EmfDescription (variable): A 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, optional): An 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. Thus, 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace2 (variable, optional)																															
...																															
EmfPixelFormat (40 bytes)																															
...																															
...																															
...																															

EmfPixelFormat (40 bytes): A 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 present in each 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 the flowchart in section 2.3.4.2 for details.

2.3.5 Drawing Record Types

The **Drawing** record types perform graphics drawing and painting functions. The clipping regions used by drawing records are maintained in the **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).

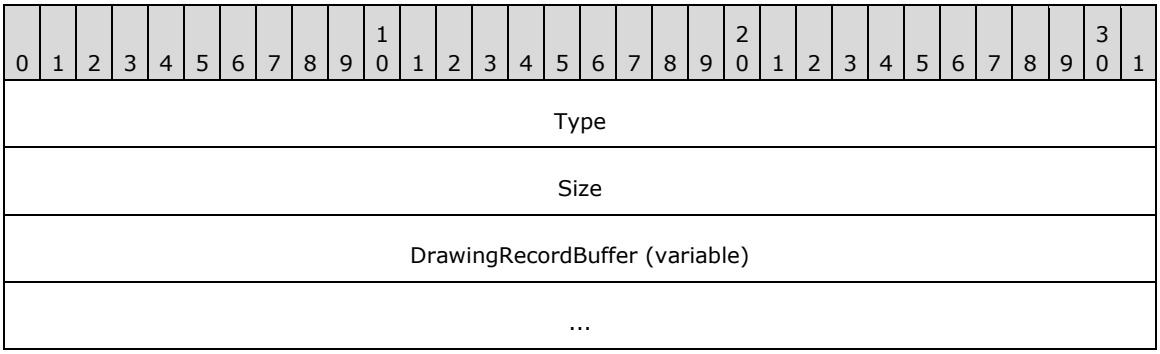
The following are EMF drawing record types.

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

Name	Section	Description
		endpoint 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 bracket and fills the path's interior with the current brush.
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.
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

Name	Section	Description
		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.
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.



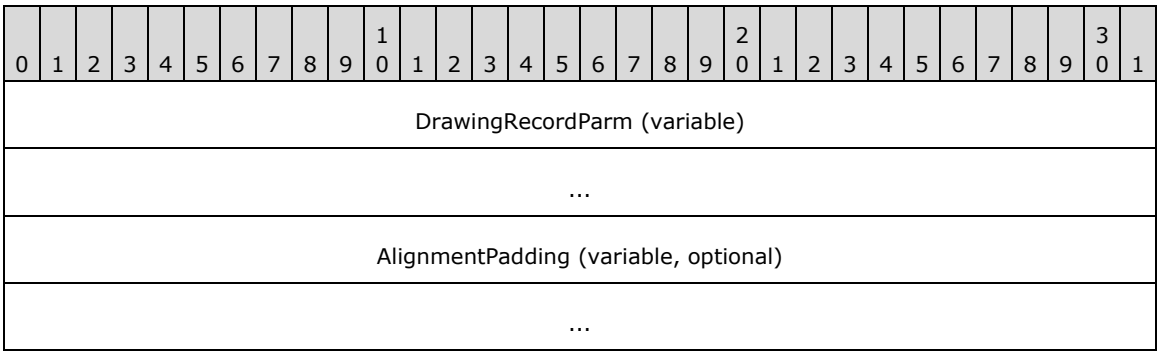
Type (4 bytes): An unsigned integer that defines the type of record. The drawing record types are listed in the following table. See the preceding table for descriptions of these records.

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

Name	Value
EMR_ELLIPSE	0x0000002A
EMR_RECTANGLE	0x0000002B
EMR_ROUNDRECT	0x0000002C
EMR_ARC	0x0000002D
EMR_CHORD	0x0000002E
EMR_PIE	0x0000002F
EMR_EXTFLOODFILL	0x00000035
EMR_LINETO	0x00000036
EMR_ARCTO	0x00000037
EMR_POLYDRAW	0x00000038
EMR_FILLPATH	0x0000003E
EMR_STROKEANDFILLPATH	0x0000003F
EMR_STROKEPATH	0x00000040
EMR_FILLRGN	0x00000047
EMR_FRAMERGN	0x00000048
EMR_PAINTRGN	0x0000004A
EMR_EXTTEXTOUTA	0x00000053
EMR_EXTTEXTOUTW	0x00000054
EMR_POLYBEZIER16	0x00000055
EMR_POLYGON16	0x00000056
EMR_POLYLINE16	0x00000057
EMR_POLYBEZIERTO16	0x00000058
EMR_POLYLINETO16	0x00000059
EMR_POLYPOLYLINE16	0x0000005A
EMR_POLYPOLYGON16	0x0000005B
EMR_POLYDRAW16	0x0000005C
EMR_POLYTEXTOUTA	0x00000060
EMR_POLYTEXTOUTW	0x00000061
EMR_SMALLTEXTOUT	0x0000006C
EMR_GRADIENTFILL	0x00000076

Size (4 bytes): An 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.



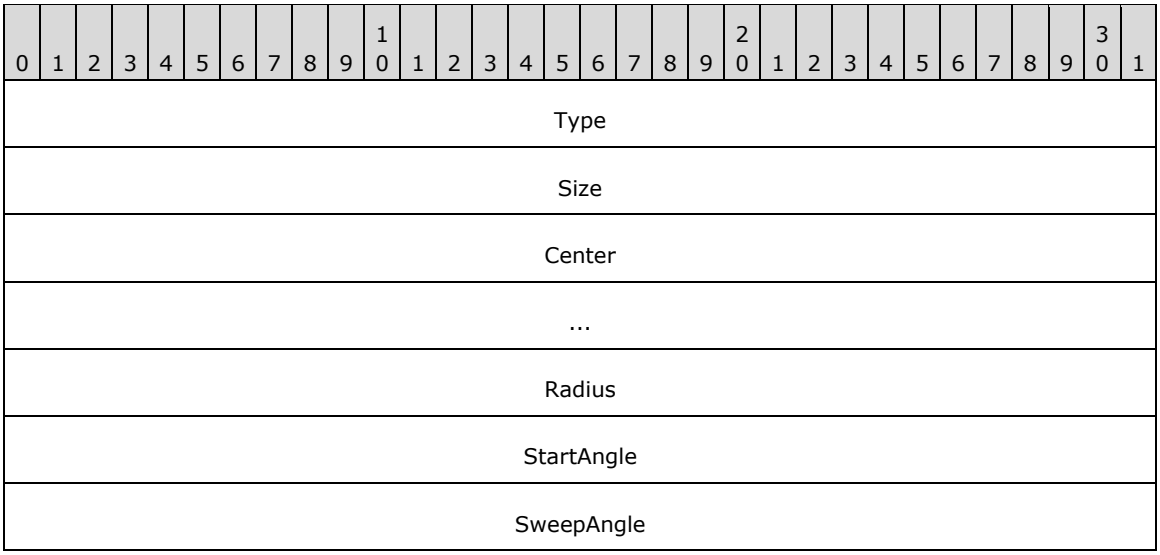
DrawingRecordParm (variable): An array of bytes that contains the parameters for the drawing record.

AlignmentPadding (variable, optional): An 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 more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ANGLEARC**. This value is 0x00000029.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Center (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the logical coordinates of the circle's center.

Radius (4 bytes): An 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.

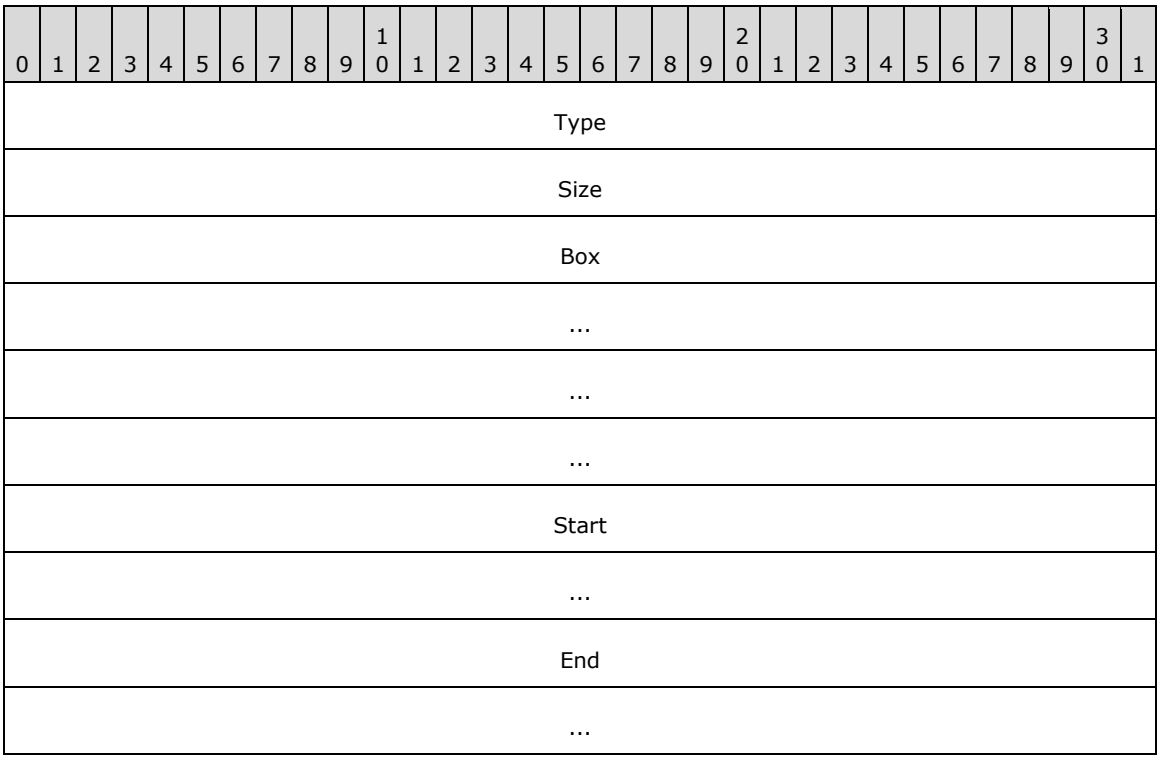
If the sweep angle is greater than 360 degrees, the arc is swept multiple times.

This record specifies lines by using the current pen. The figure is not filled.

See section 2.3.5 for more drawing record types.

2.3.5.2 EMR_ARC Record

The EMR_ARC record specifies an elliptical arc.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ARC**. This value is 0x0000002D.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle in logical units.

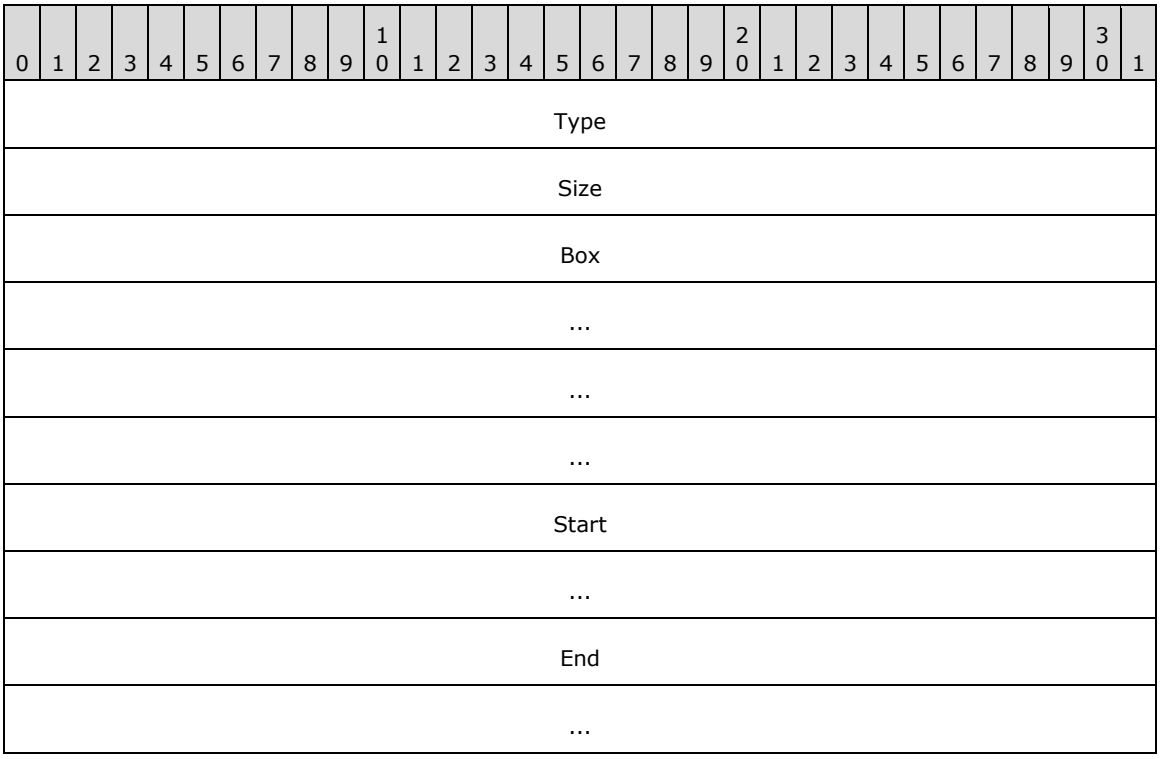
Start (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the coordinates in logical units of the ending point of the radial line defining the starting point of the arc.

End (8 bytes): A 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 more drawing record types.

2.3.5.3 EMR_ARCTO Record

The EMR_ARCTO record specifies an elliptical arc. It resets the current drawing position to the endpoint of the arc.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ARCTO**. This value is 0x00000037.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle in logical units.

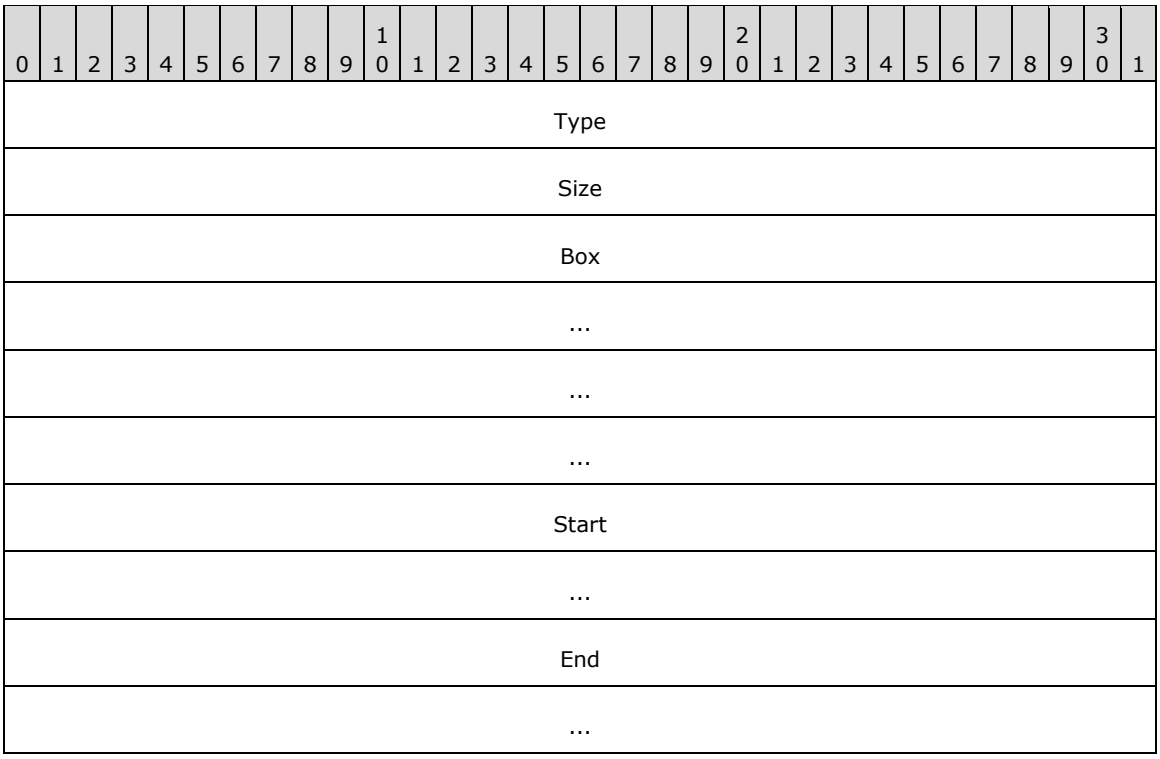
Start (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the coordinates, in logical units, of the first radial ending point, in logical units.

End (8 bytes): A PointL object that specifies the coordinates of the second radial ending point, in logical units.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CHORD**. This value is 0x0000002E.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle in logical units.

Start (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the coordinates, in logical units, of the endpoint of the radial defining the beginning of the chord.

End (8 bytes): A 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.

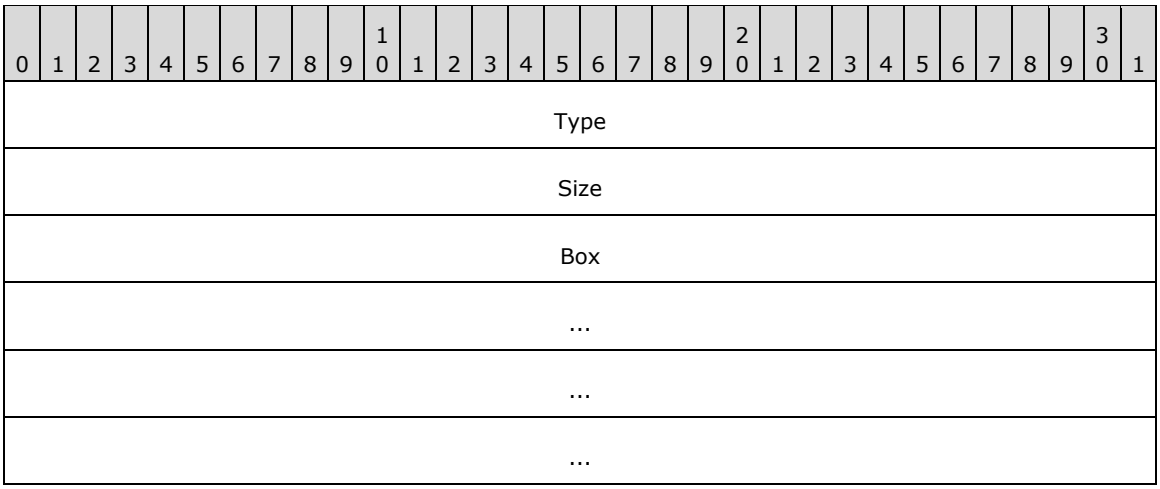
If the starting point and ending point of the curve are the same, a complete ellipse is drawn.

The current drawing position is neither used nor updated by processing this record.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ELLIPSE**. This value is 0x0000002A.

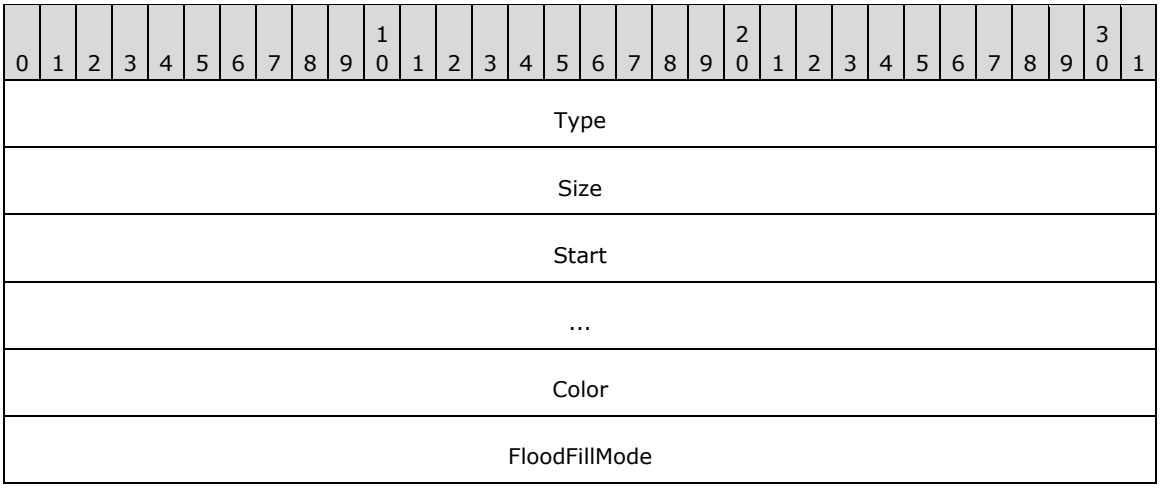
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle in logical units.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_EXTFLOODFILL**. This value is 0x00000035.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Start (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the coordinates, in logical units, where filling begins.

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

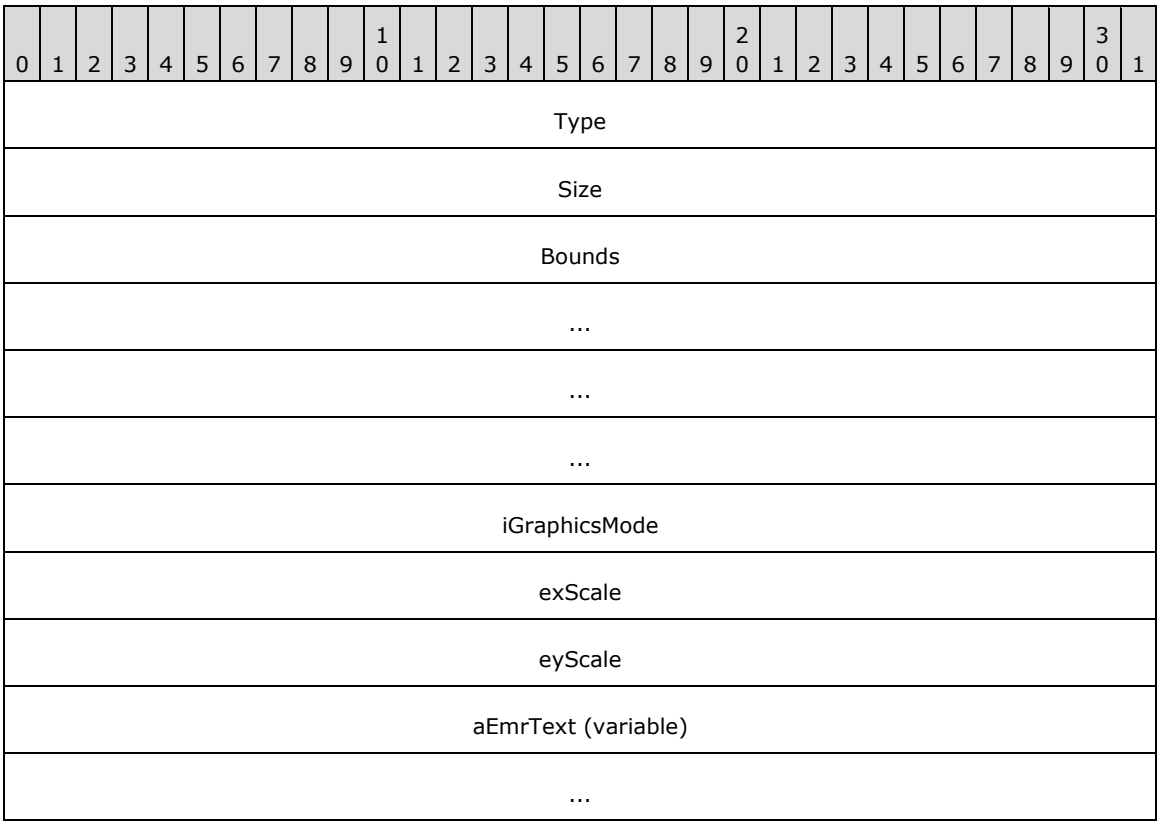
FloodFillMode (4 bytes): An unsigned integer that specifies how to use the **Color** value to determine the area for the flood fill operation. This value is in the FloodFill enumeration (section 2.1.13).

See section 2.3.5 for more drawing record types.

2.3.5.7 EMR_EXTTEXTOUTA Record

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

Fields not specified in this section are specified in section 2.3.5.



Type (4 bytes): An unsigned integer that identifies the record type as EMR_EXTTEXTOUTA from the RecordType enumeration (section 2.1.1). This value is 0x00000053.

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

iGraphicsMode (4 bytes): An unsigned integer that specifies the graphics mode from the GraphicsMode enumeration (section 2.1.16).

exScale (4 bytes): A FLOAT 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 FLOAT 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 the state of the current graphics environment (section 3.1.1.2). A rectangle for clipping and/or opaquing can be defined in the EmrText object in the **aEmrText** field.

This record SHOULD<64> be emulated with an EMR_EXTTEXTOUTW record (section 2.3.5.8), which requires the ASCII text string in the EmrText object to be converted to Unicode UTF16-LE encoding.

See section 2.3.5 for more 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.

Fields not specified in this section are specified in section 2.3.5.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
wEmrText (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies the record type as **EMR_EXTTEXTOUTW** from the RecordType enumeration (section 2.1.1). This value is 0x00000054.

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

iGraphicsMode (4 bytes): An unsigned integer that specifies the current graphics mode from the GraphicsMode enumeration (section 2.1.16).

exScale (4 bytes): A FLOAT 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 FLOAT 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**.

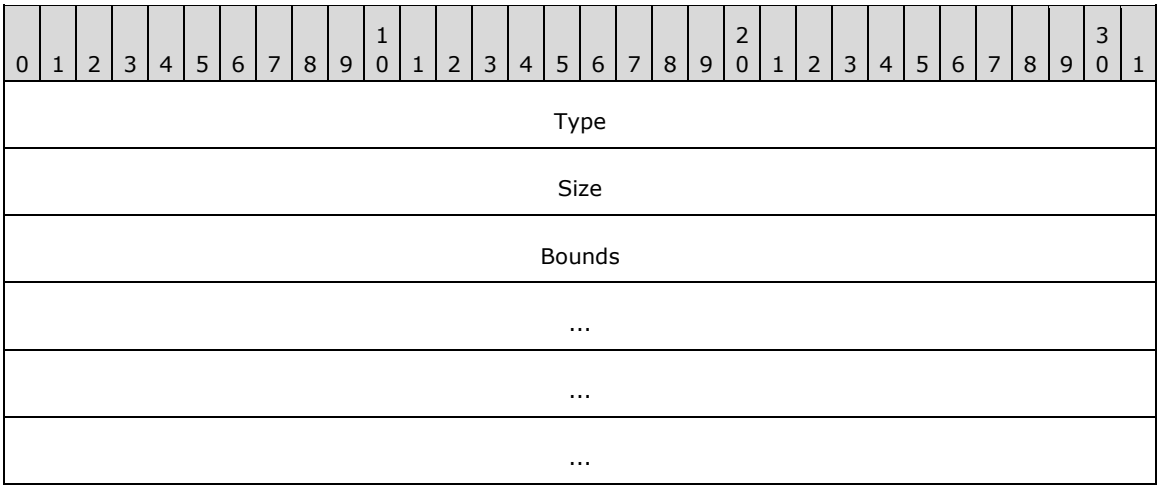
wEmrText (variable): An EmrText object (section 2.2.5) that specifies the output string in Unicode UTF16-LE characters, with text attributes and spacing values.

The font and text colors used for output are specified by properties in the current state of EMF metafile playback (section 3.1). A rectangle for clipping and/or opaquing can be defined in the EmrText object that is specified in the **aEmrText** field.

See section 2.3.5 for more drawing record types.

2.3.5.9 EMR_FILLPATH Record

The EMR_FILLPATH record closes any open figures in the current path bracket and fills the path's interior by using the current brush and polygon-filling mode.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_FILLPATH**. This value is 0x0000003E.

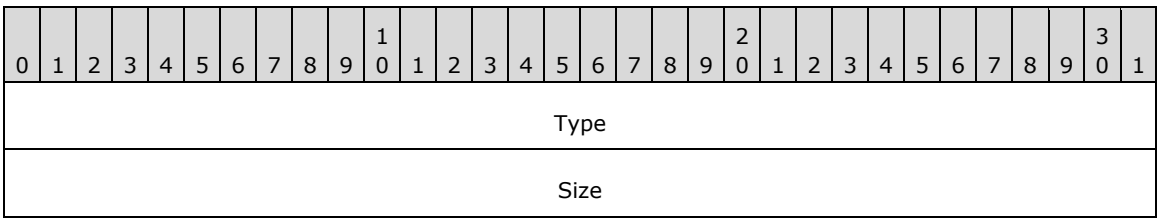
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

See section 2.3.5 for more drawing record types.

2.3.5.10 EMR_FILLRGN Record

The EMR_FILLRGN record fills the specified region by using the specified brush. The current clipping regions used by this record are maintained in a **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).



Bounds
...
...
...
RgnDataSize
ihBrush
RgnData (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_FILLRGN**. This value is 0x00000047.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical units. If the intersection of this rectangle with the current clipping region is empty, this record has no effect.

RgnDataSize (4 bytes): An unsigned integer that specifies the size of region data in bytes.

ihBrush (4 bytes): An unsigned integer that specifies the index of the brush in the EMF object table (section 3.1.1.1) for filling the region.

RgnData (variable): A **RgnDataSize** length array of bytes that specifies the output region in a **RegionData** object (section 2.2.24). The bounds specified by the **RegionDataHeader** field of this object MAY<65> be used as the bounding region when this record is processed.

See section 2.3.5 for more 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. The current clipping regions used by this record are maintained in a **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															

...
...
RgnDataSize
ihBrush
Width
Height
RgnData (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_FRAMERGN**. This value is 0x00000048.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping region is empty, this record has no effect.

RgnDataSize (4 bytes): An unsigned integer that specifies the size of region data in bytes.

ihBrush (4 bytes): An unsigned integer that specifies the index of the brush in the EMF object table index.

Width (4 bytes): A signed integer that specifies the width of the vertical brush stroke, in logical units.

Height (4 bytes): A signed integer that specifies the height of the horizontal brush stroke, in logical units.

RgnData (variable): A **RgnDataSize** length array of bytes that specifies the output region in a **RegionData** object (section 2.2.24). The bounds specified by the **RegionDataHeader** field of this object MAY<66> be used as the bounding region when this record is processed.

See section 2.3.5 for more drawing record types.

2.3.5.12 EMR_GRADIENTFILL Record

The **EMR_GRADIENTFILL** record specifies filling rectangles or triangles with gradients of color.<67>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															

...
...
...
nVer
nTri
ulMode
VertexData (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_GRADIENTFILL. This value is 0x00000076.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

nVer (4 bytes): An unsigned integer that specifies the number of vertexes.

nTri (4 bytes): An unsigned integer that specifies the number of rectangles or triangles to fill.

ulMode (4 bytes): An unsigned integer that specifies the gradient fill mode. This value is 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
VertexObjects (variable)																															
...																															
VertexIndexes (variable)																															
...																															
VertexPadding (variable, optional)																															
...																															

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, optional): An 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 **ulMode** 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.<68>

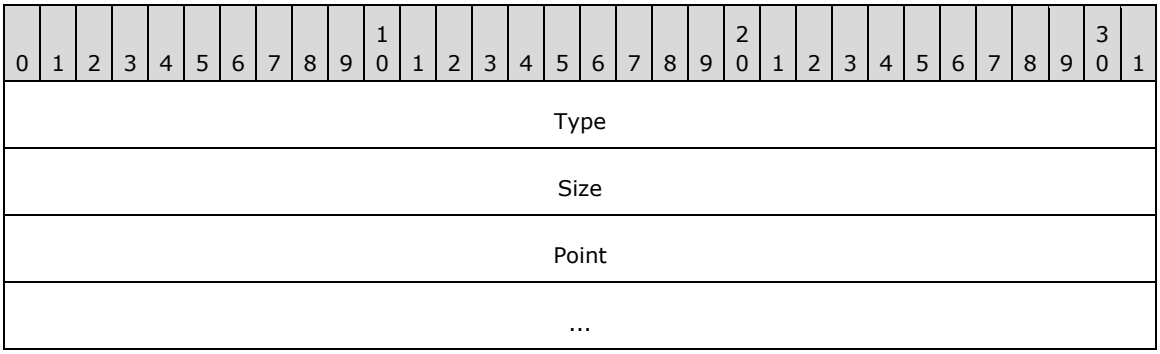
An EMR_GRADIENTFILL record that specifies that the upper-left and lower-right vertexes of a rectangle SHOULD fill the figure with smooth gradients of color. There are two gradient fill modes in the GradientFill enumeration that can be used when drawing a rectangle. In **GRADIENT_FILL_RECT_H** mode, the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

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 more drawing record types.

2.3.5.13 EMR_LINETO Record

The EMR_LINETO record specifies a line from the current drawing position up to, but not including, the specified point. It resets the current position to the specified point.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_LINETO**. This value is 0x00000036.

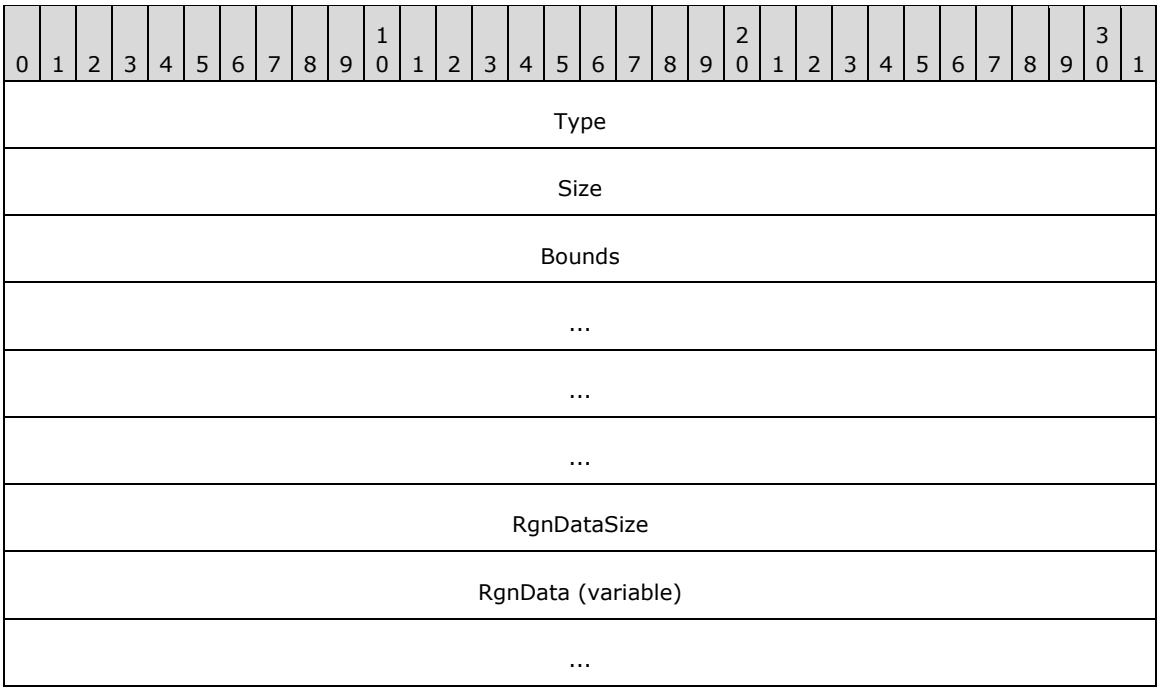
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Point (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the coordinates of the line's endpoint.

See section 2.3.5 for more drawing record types.

2.3.5.14 EMR_PAINTRGN Record

The EMR_PAINTRGN record paints the specified region by using the current brush. The current clipping regions used by this record are maintained in a **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_PAINTRGN**. This value is 0x0000004A.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping region is empty, this record has no effect.

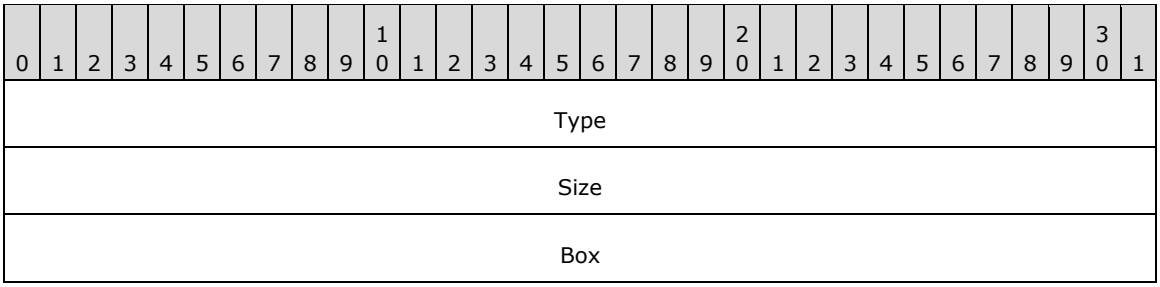
RgnDataSize (4 bytes): An unsigned integer that specifies the size of the **RgnData** field data in bytes.

RgnData (variable): An array of bytes that specifies the output region in a **RegionData** object (section 2.2.24). The bounds specified by the **RegionDataHeader** field of that object MAY<69> be used as the bounding rectangle of the region when this record is processed.

See section 2.3.5 for more 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.



...
...
...
Start
...
End
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_PIE**. This value is 0x0000002F.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle in logical units.

Start (8 bytes): A PointL object ([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 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 drawing position is neither used nor updated by this record.

See section 2.3.5 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															

...
Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYBEZIER**. This value is 0x00000002.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

Count (4 bytes): An 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): An array of PointL objects ([MS-WMF] section 2.2.2.15) that specify the endpoints and control points of the Bezier curves in logical units.

Cubic Bezier curves are defined using the endpoints and control points specified by the **aPoints** field. The first curve is drawn from the first point to the fourth point, using the second and third points as control points. Each subsequent curve in the sequence needs exactly three more points: the ending point of the previous curve is used as the starting point, the next two points in the sequence are control points, and the third is the ending point.

The cubic Bezier curves SHOULD be drawn using the current pen.

See section 2.3.5 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															

Bounds
...
...
...
Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYBEZIER16**. This value is 0x00000055.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

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

Count (4 bytes): An 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): An array of **PointS** objects ([MS-WMF] section 2.2.2.16), which specify the points of the Bezier curves in logical units.

See section 2.3.5 for more drawing record types.

2.3.5.18 EMR_POLYBEZIERTO Record

The EMR_POLYBEZIERTO record specifies one or more Bezier curves based upon the current drawing position.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYBEZIERTO**. This value is 0x00000005.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

Count (4 bytes): An 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): An array of PointL objects ([MS-WMF] section 2.2.2.15), which specify the endpoints and control points of the Bezier curves in logical units.

The Bezier curves SHOULD be drawn using the current pen.

See section 2.3.5 for more drawing record types.

2.3.5.19 EMR_POLYBEZIERTO16 Record

The EMR_POLYBEZIERTO16 record specifies one or more Bezier curves based on the current drawing position.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYBEZIERTO16**. This value is 0x00000058.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

Count (4 bytes): An 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): An array of **PointS** objects ([MS-WMF] section 2.2.2.16), which specify the points of the Bezier curves in logical units.

See section 2.3.5 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															
abTypes (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYDRAW**. This value is 0x00000038.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

Count (4 bytes): An unsigned integer that specifies the number of points in the **aPoints** field.

aPoints (variable): An array of PointL objects ([MS-WMF] section 2.2.2.15), which specify the 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 is in the Point (section 2.1.26) enumeration.

See section 2.3.5 for more drawing record types.

2.3.5.21 EMR_POLYDRAW16 Record

The EMR_POLYDRAW16 record specifies a set of line segments and Bezier curves.

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYDRAW16**. This value is 0x0000005C.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

Count (4 bytes): An unsigned integer that specifies the number of points.

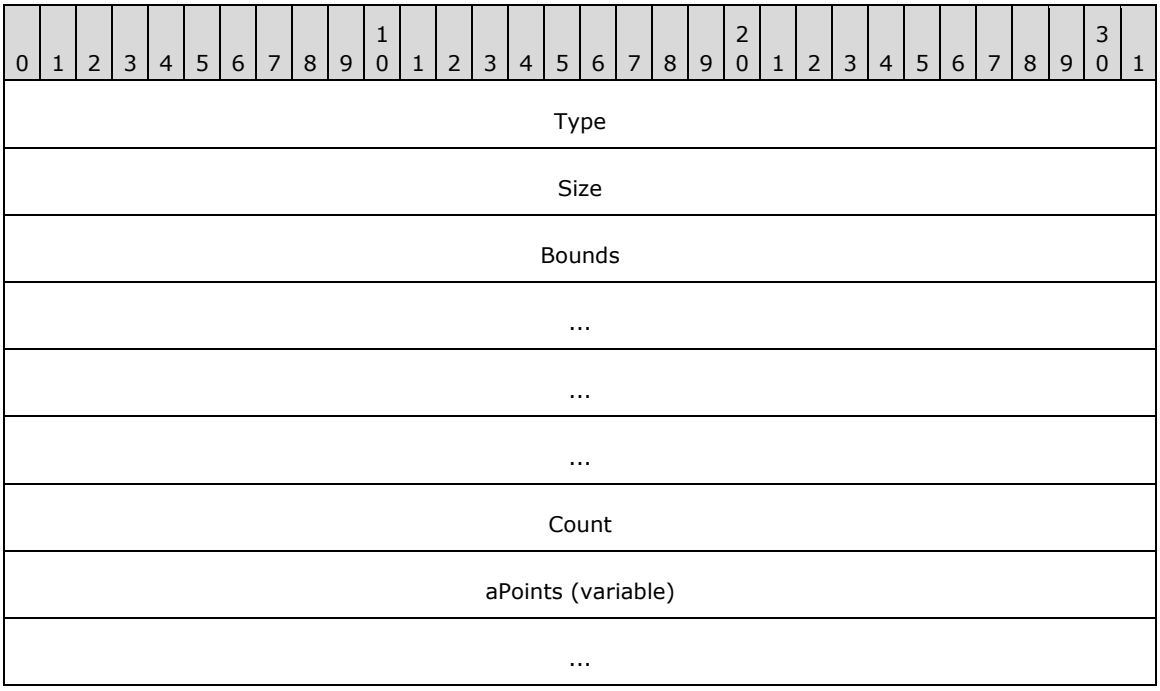
aPoints (variable): A **Count** length array of **PointS** objects, specified in [MS-WMF] section 2.2.2.16, which specifies the array of points.

abTypes (variable): A **Count** length array of bytes that specifies the point types. This value is in the Point (section 2.1.26) enumeration.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYGON**. This value is 0x00000003.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

Count (4 bytes): An 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 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 more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYGON16**. This value is 0x00000056.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

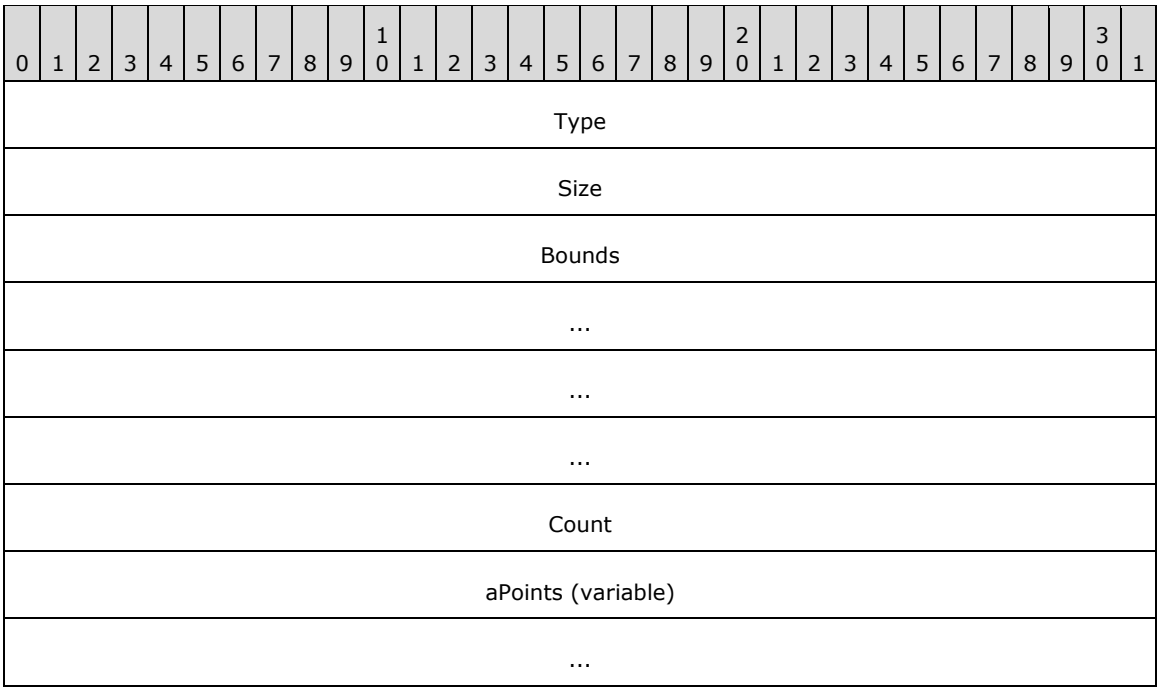
Count (4 bytes): An unsigned integer that specifies the total number of points.

aPoints (variable): A **Count** length array of **PointS** objects, specified in [MS-WMF] section 2.2.2.16, which specifies the array of points.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYLINE**. This value is 0x00000004.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

Count (4 bytes): An 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.

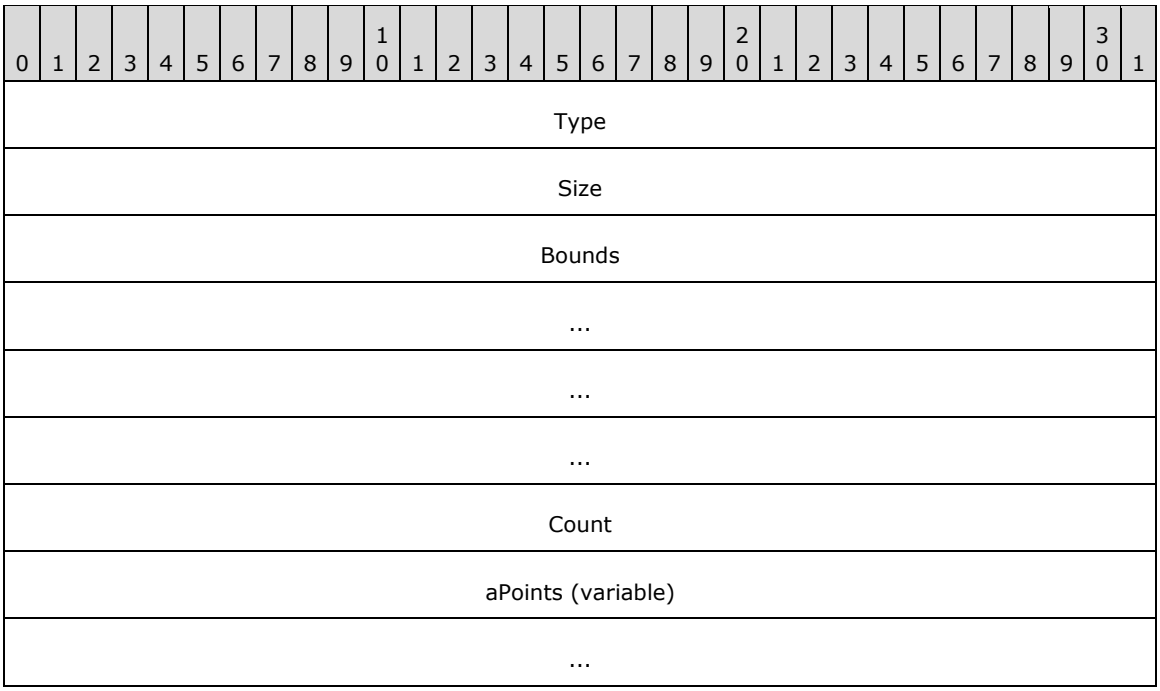
aPoints (variable): A **Count** length array of PointL objects ([MS-WMF] section 2.2.2.15) that specifies the point data, in logical units.

The line segments SHOULD be drawn using the current pen.

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYLINE16**. This value is 0x00000057.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

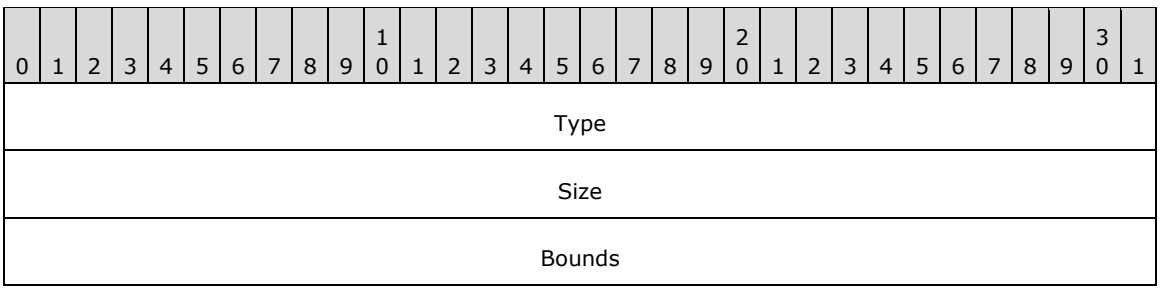
Count (4 bytes): An unsigned integer that specifies the total number of points.

aPoints (variable): A **Count** length array of **PointS** objects, specified in [MS-WMF] section 2.2.2.16, which specifies the array of points.

See section 2.3.5 for more drawing record types.

2.3.5.26 EMR_POLYLINETO Record

The EMR_POLYLINETO record specifies one or more straight lines based upon the current drawing position.



...
...
...
Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYLINETO**. This value is 0x00000006.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

Count (4 bytes): An 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 PointL objects ([MS-WMF] section 2.2.2.15, which 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 more drawing record types.

2.3.5.27 EMR_POLYLINETO16 Record

The EMR_POLYLINETO16 record specifies one or more straight lines based upon the current drawing position.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															

...
...
...
Count
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYLINETO16**. This value is 0x00000059.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

Count (4 bytes): An unsigned integer that specifies the number of points.

aPoints (variable): A **Count** length array of **PointS** objects ([MS-WMF] section 2.2.2.16, which specifies the array of points.

A line is drawn from the current drawing 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**.

See section 2.3.5 for more drawing record types.

2.3.5.28 EMR_POLYPOLYGON Record

The EMR_POLYPOLYGON record specifies a series of closed polygons.

Fields not specified in this section are specified in section 2.3.5.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
NumberOfPolygons																															

Count
PolygonPointCount (variable)
...
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYPOLYGON**. This value is 0x00000008.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

NumberOfPolygons (4 bytes): An unsigned integer that specifies the number of polygons.

Count (4 bytes): An 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 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 more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															

Size
Bounds
...
...
...
NumberOfPolygons
Count
PolygonPointCount (variable)
...
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYPOLYGON16**. This value is 0x0000005B.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

NumberOfPolygons (4 bytes): An unsigned integer that specifies the number of polygons.

Count (4 bytes): An 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 **PointS** objects ([MS-WMF] section 2.2.2.16), which specifies the array of points.

See section 2.3.5 for more drawing record types.

2.3.5.30 EMR_POLYPOLYLINE Record

The EMR_POLYPOLYLINE record draws multiple series of connected line segments.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															

Bounds
...
...
...
NumberOfPolylines
Count
aPolylinePointCount (variable)
...
aPoints (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYPOLYLINE**. This value is 0x00000007.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

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

Count (4 bytes): An 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 >= 0x00000002.

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

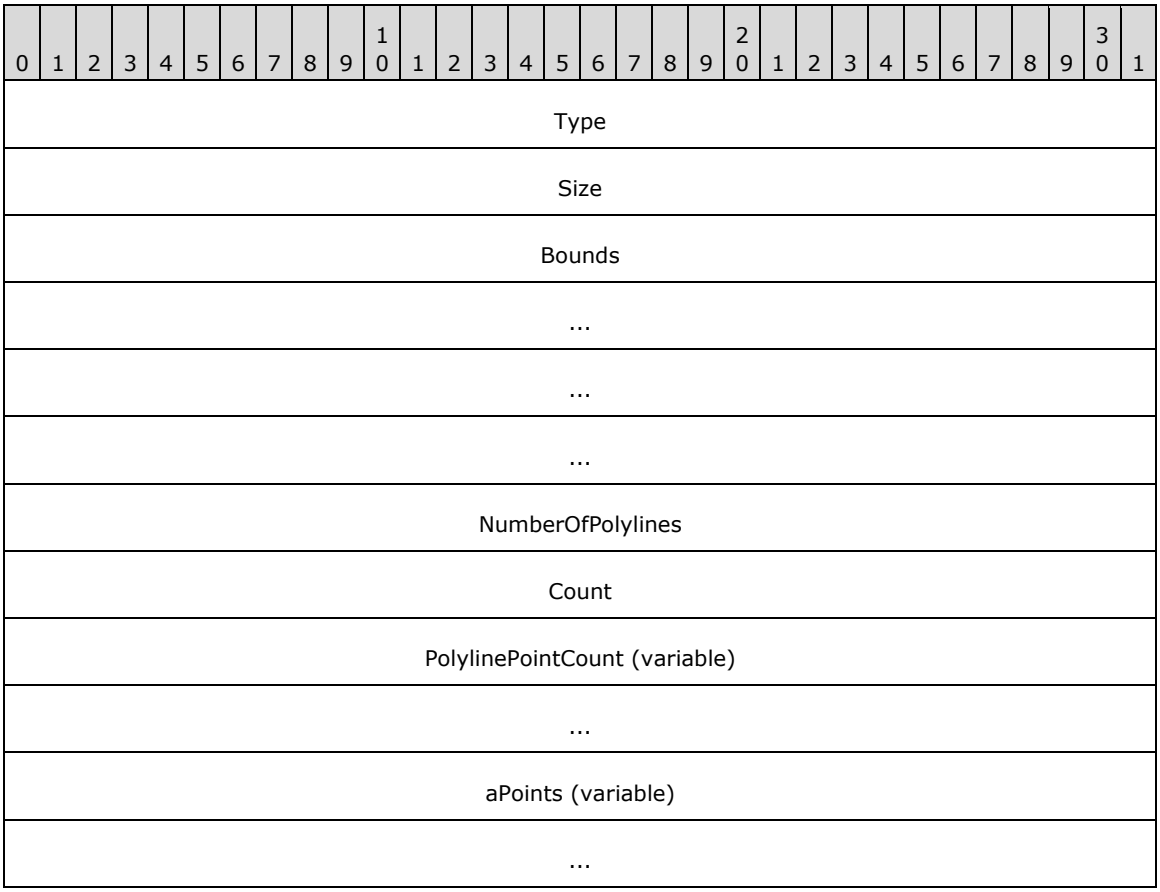
aPoints (variable): A **Count**-length array of PointL objects ([MS-WMF] section 2.2.2.15) that specify the point data, in logical units.

The line segments SHOULD be drawn using the current pen. The figures formed by the segments SHOULD NOT filled. The current drawing position SHOULD neither be used nor updated by this record.

See section 2.3.5 for more drawing record types.

2.3.5.31 EMR_POLYPOLYLINE16 Record

The EMR_POLYPOLYLINE16 record specifies multiple series of connected line segments.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYPOLYLINE16**. This value is 0x0000005A.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

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

NumberOfPolylines (4 bytes): An unsigned integer that specifies the number of polylines.

Count (4 bytes): An 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.

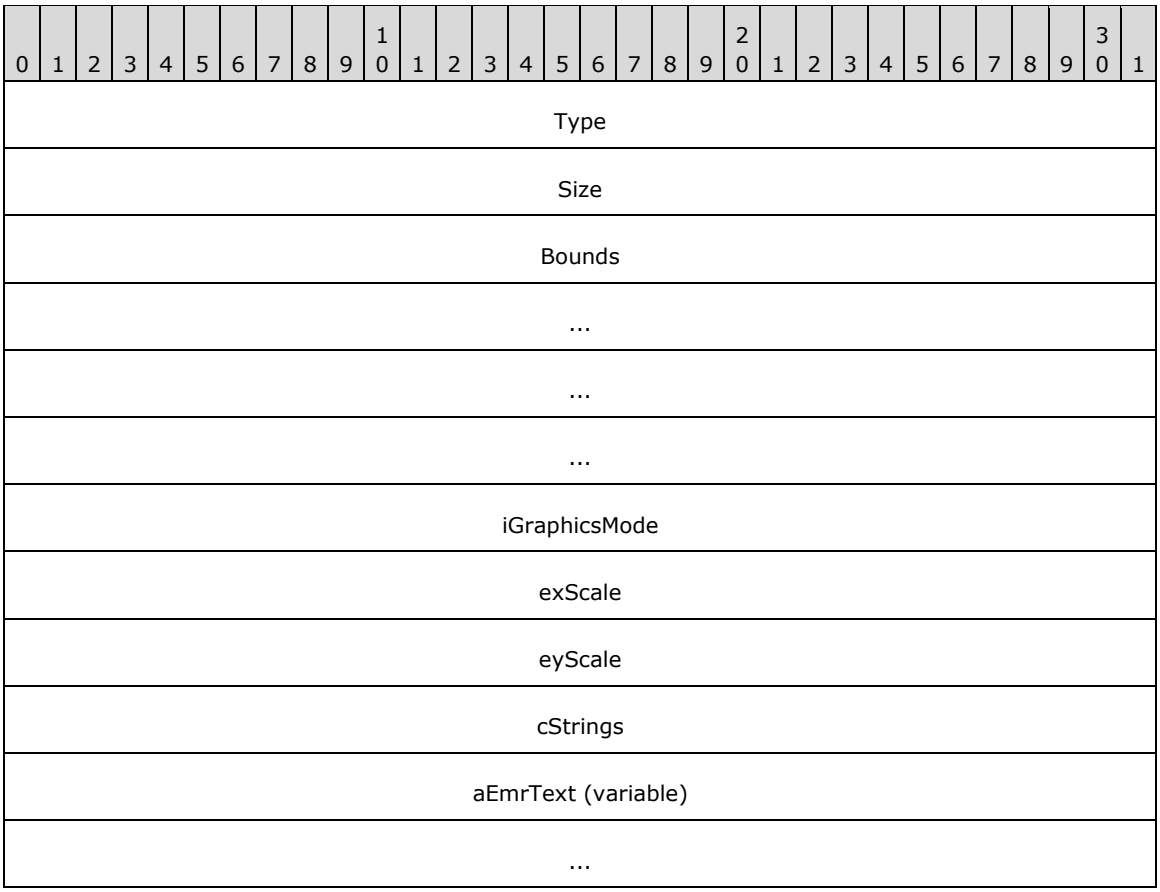
aPoints (variable): A **Count** length array of **PointS** objects ([MS-WMF] section 2.2.2.16), which specifies the array of points.

See section 2.3.5 for more 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.

Fields not specified in this section are specified in section 2.3.5.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYTEXTOUTA**. This value is 0x00000060.

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

iGraphicsMode (4 bytes): An unsigned integer that specifies the current graphics mode, from the GraphicsMode enumeration (section 2.1.16).

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

eyScale (4 bytes): A FLOAT value that specifies the Y scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

cStrings (4 bytes): An 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 EMF metafile playback (section 3.1).

EMR_POLYTEXTOUTA SHOULD<70> 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.

See section 2.3.5 for more 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.

Fields not specified in this section are specified in section 2.3.5.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
cStrings																															
wEmrText (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_POLYTEXTOUTW**. This value is 0x00000061.

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

iGraphicsMode (4 bytes): An unsigned integer that specifies the current graphics mode. Graphics modes are specified in section 2.1.16.

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

eyScale (4 bytes): A FLOAT value that specifies the X scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

cStrings (4 bytes): An 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 Unicode UTF16-LE characters, with text attributes and spacing values. The number of EmrText objects is specified by **cStrings**.

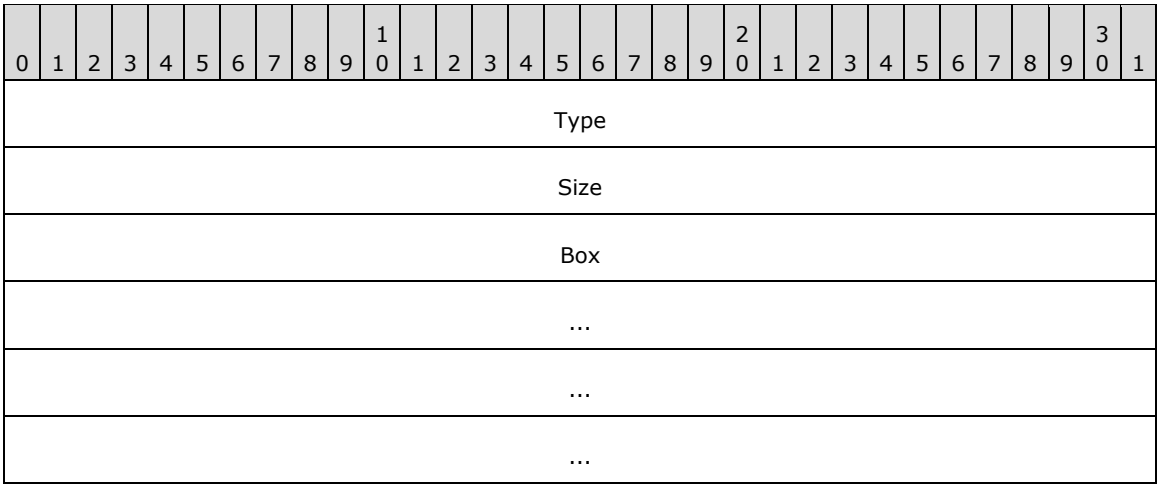
The font and text colors used for output are specified by properties in the current state of the playback device context.

EMR_POLYTEXTOUTW SHOULD be emulated with a series of EMR_EXTTEXTOUTW records (section 2.3.5.7), one per string.<71>

See section 2.3.5 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_RECTANGLE**. This value is 0x0000002B.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive rectangle to draw.

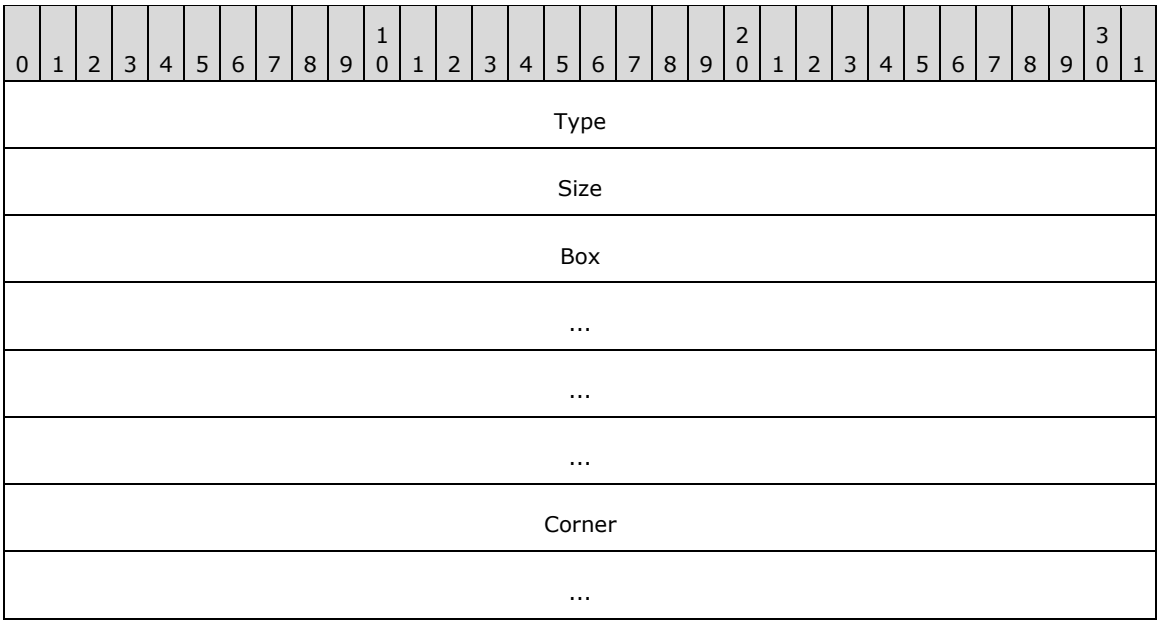
The current drawing position is neither used nor updated by this record.

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 more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_ROUNDRECT**. This value is 0x0000002C.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

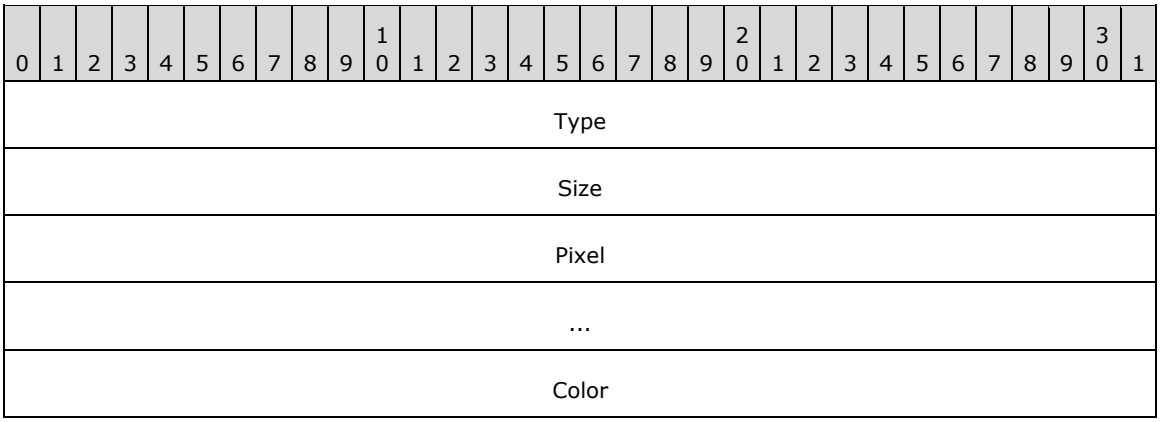
Box (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19), which specifies the inclusive-inclusive bounding rectangle, in logical coordinates.

Corner (8 bytes): A 64-bit **SizeL** object ([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 more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETPIXELV**. This value is 0x0000000F.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Pixel (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15) that specifies the logical coordinates for the pixel.

Color (4 bytes): A 32-bit ColorRef object ([MS-WMF] section 2.2.2.8) that specifies the pixel color.

See section 2.3.5 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
x																															
y																															
cChars																															
fuOptions																															
iGraphicsMode																															
exScale																															
eyScale																															
Bounds (optional)																															
...																															
...																															
...																															
TextString (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SMALLTEXTOUT**. This value is 0x0000006C.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

x (4 bytes): A signed integer specifying the x-coordinate of where to place the string.

y (4 bytes): A signed integer specifying the y-coordinate of where to place the string.

cChars (4 bytes): An unsigned integer specifying the number of 16-bit characters in the string. The string is NOT null-terminated.

fuOptions (4 bytes): An 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): An unsigned integer specifying the graphics mode, from the GraphicsMode enumeration (section 2.1.16).

exScale (4 bytes): A FLOAT value that specifies how much to scale the text in the x-direction.

eyScale (4 bytes): A FLOAT value that specifies how much to scale the text in the y-direction.

Bounds (16 bytes, optional): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

TextString (variable): A 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 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 more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_STROKEANDFILLPATH**. This value is 0x0000003F.

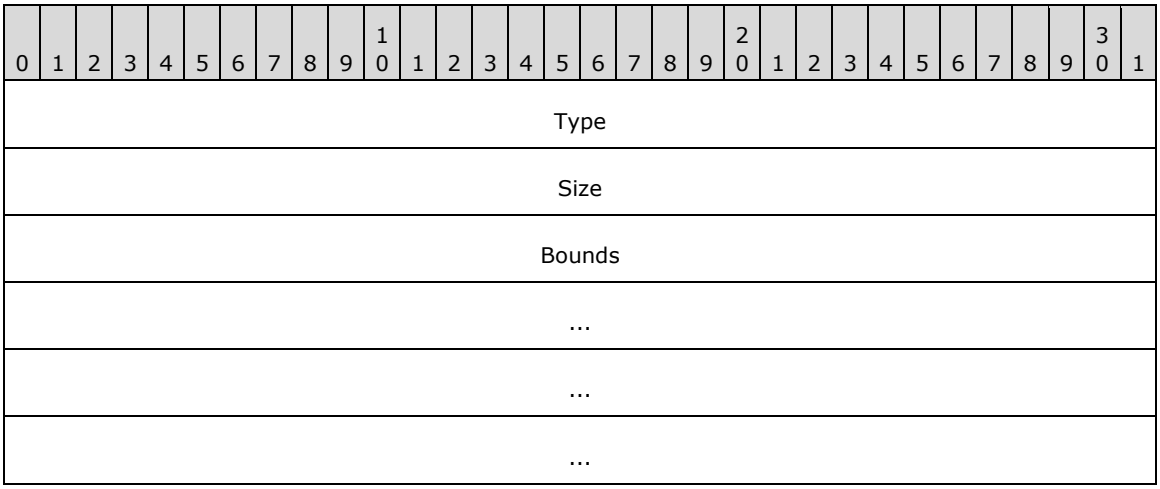
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

See section 2.3.5 for more drawing record types.

2.3.5.39 EMR_STROKEPATH Record

The EMR_STROKEPATH record renders the specified path by using the current pen.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_STROKEPATH**. This value is 0x00000040.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the bounding rectangle in logical units.

See section 2.3.5 for more drawing record types.

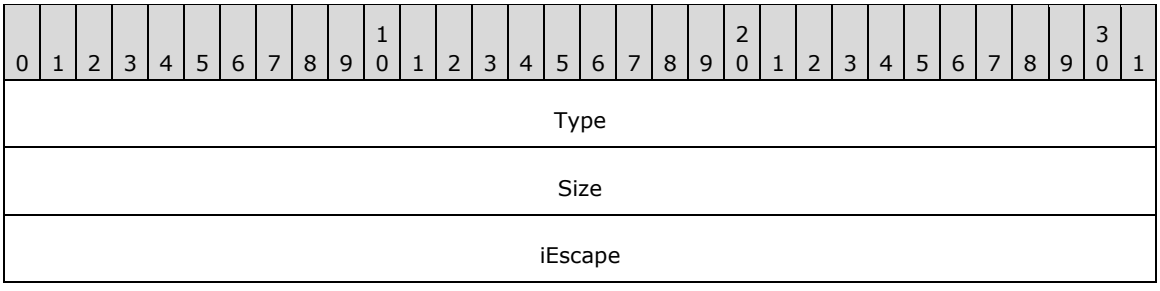
2.3.6 Escape Record Types

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 results in drawing being done.
EMR_EXTESCAPE	2.3.6.2	Passes arbitrary information to the printer driver. The intent is that the information does 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.



EscapeRecordBuffer (variable)
...

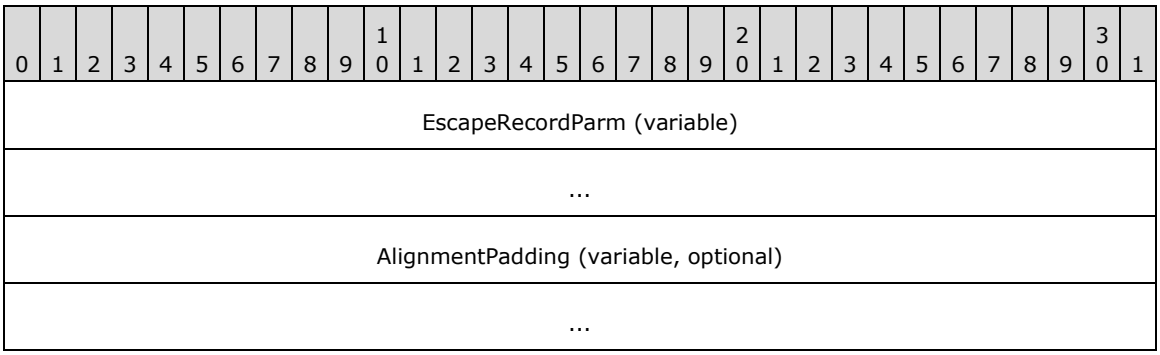
Type (4 bytes): An unsigned integer that defines the type of the record. The escape record types are listed in the following table. See the preceding table for descriptions of these records.

Name	Value
EMR_DRAWESCAPE	0x00000069
EMR_EXTESCAPE	0x0000006A
EMR_NAMEDESCAPE	0x0000006E

Size (4 bytes): An 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): An unsigned integer that specifies the printer driver escape to execute. This MUST be one of the values in the MetafileEscapes enumeration ([MS-WMF] section 2.1.1.17).

EscapeRecordBuffer (variable): An array of bytes that contains the remainder of the escape record.



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

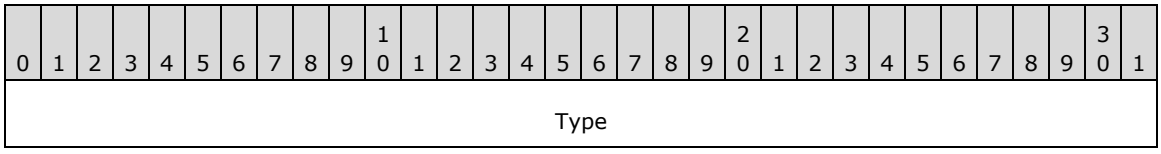
AlignmentPadding (variable, optional): An array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section 2.3 for more 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 results in drawing being done.

Fields not specified in this section are specified in section 2.3.6.



Size
iEscape
cjIn
Data (variable)
...

Type (4 bytes): An 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): An 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 more 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 does not result in drawing being done.

Fields not specified in this section are specified in section 2.3.6.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
iEscape																															
cjIn																															
Data (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type from the EmrComment enumeration (section 2.1.10). This value is 0x0000006A.

cjIn (4 bytes): An unsigned integer specifying the number of bytes to pass to the printer driver.

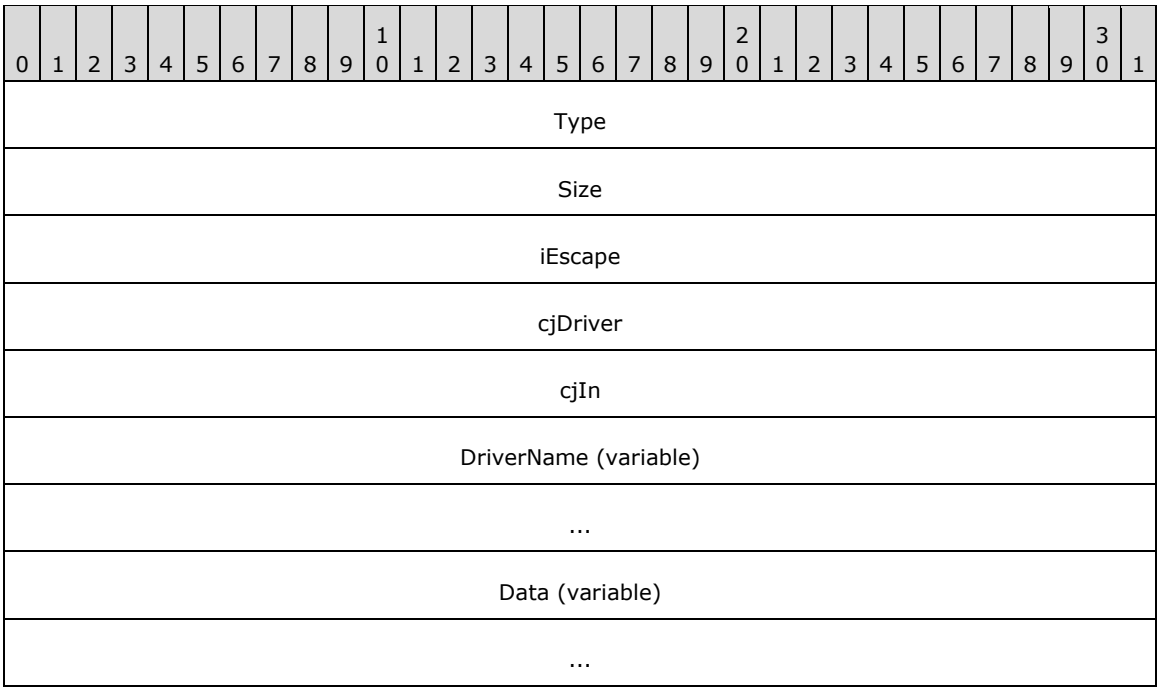
Data (variable): The data to pass to the printer driver. There MUST be **cjIn** bytes available.

See section 2.3.6 for more escape record types.

2.3.6.3 EMR_NAMEDESCAPE Record

The **EMR_NAMEDESCAPE** record passes arbitrary information to a named printer driver.

Fields not specified in this section are specified in section 2.3.6.



Type (4 bytes): An 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): An unsigned integer that specifies the number of bytes in the **DriverName** field. This value MUST be an even number.

cjIn (4 bytes): An unsigned integer specifying the number of bytes in the **Data** field to pass to the printer driver.

DriverName (variable): A null-terminated string of Unicode characters that specifies the name of the printer driver to receive data.

Data (variable): The data to pass to the printer driver.

See section 2.3.6 for more escape record types.

2.3.7 Object Creation Record Types

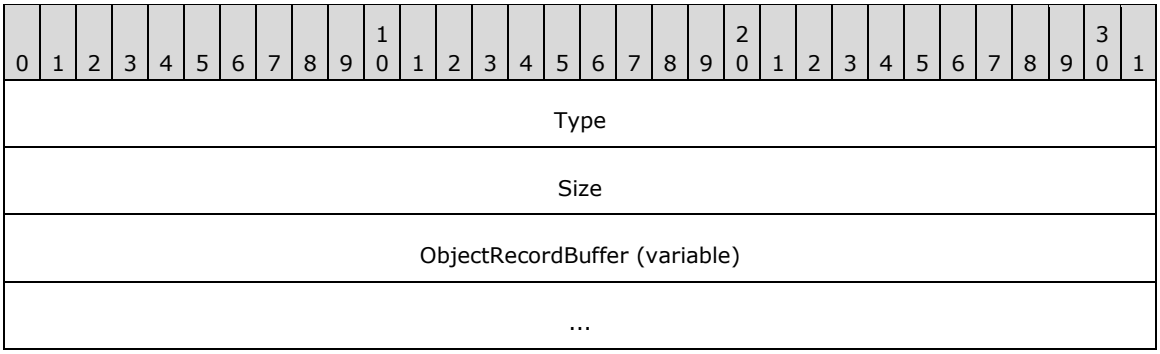
The **Object Creation** record types create graphics objects.

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 LogColorSpace object ([MS-WMF] section 2.2.2.11).
EMR_CREATECOLORSPACEW	2.3.7.3	Defines a logical color space with a LogColorSpaceW object ([MS-WMF] section 2.2.2.12).

Name	Section	Description
EMR_CREATEDIBPATTERNBRUSHPT	2.3.7.4	Defines a pattern brush with a DeviceIndependentBitmap object ([MS-WMF] section 2.2.2.9).
EMR_CREATEMONOBRUSH	2.3.7.5	Defines a monochrome pattern brush with a monochrome 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 object (section 2.2.13) or LogFontExDv object (section 2.2.15).
EMR_EXTCREATEPEN	2.3.7.9	Defines a logical pen with a LogPenEx object (section 2.2.20) and optional DeviceIndependentBitmap object.

The generic structure of object creation records is specified as follows.

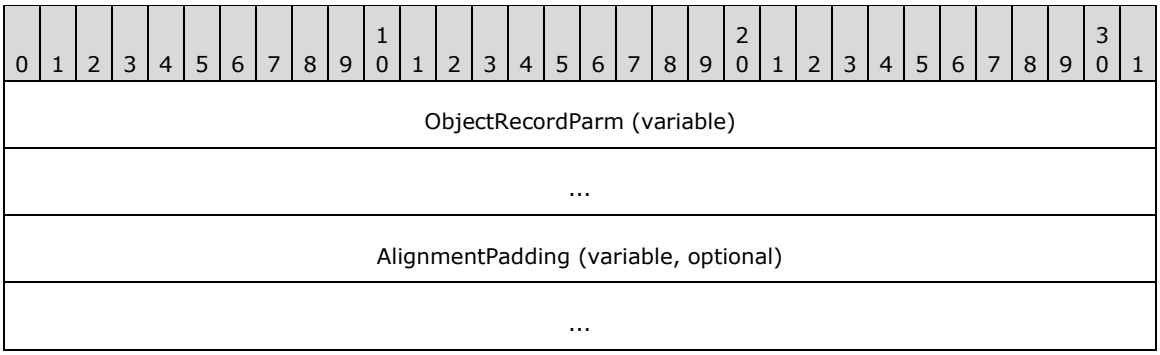


Type (4 bytes): An unsigned integer that defines the type of record. The object creation record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_CREATEMONOBRUSH	0x0000005D
EMR_CREATEDIBPATTERNBRUSHPT	0x0000005E
EMR_EXTCREATEPEN	0x0000005F
EMR_CREATECOLORSPACEW	0x0000007A
EMR_CREATEPEN	0x00000026
EMR_CREATEBRUSHINDIRECT	0x00000027
EMR_CREATEPALETTE	0x00000031
EMR_EXTCREATEFONTINDIRECTW	0x00000052
EMR_CREATECOLORSPACE	0x00000063

Size (4 bytes): An 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.



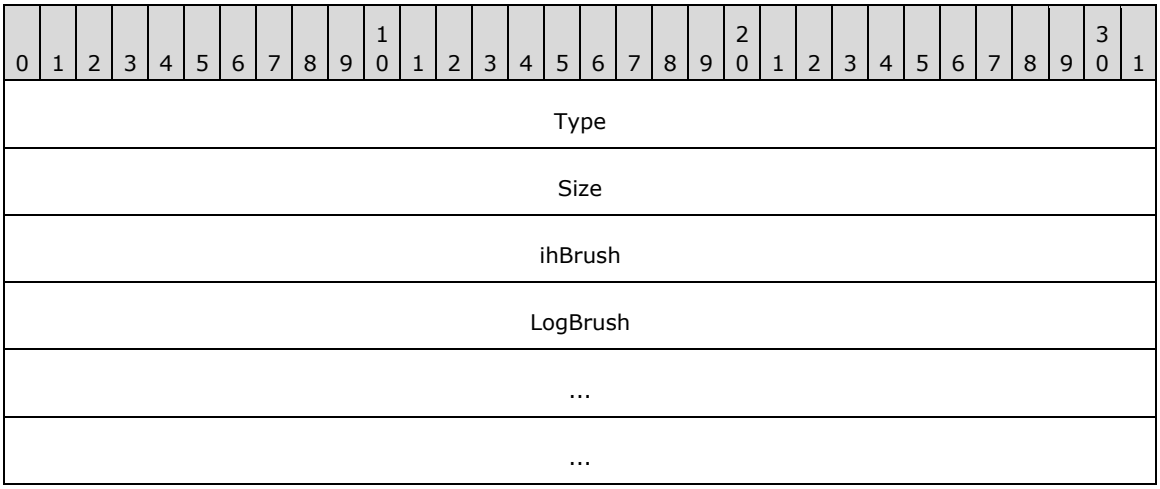
ObjectRecordParm (variable): An array of bytes that contains the parameters for the object creation record.

AlignmentPadding (variable, optional): An 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 more EMF record types.

2.3.7.1 EMR_CREATEBRUSHINDIRECT Record

The **EMR_CREATEBRUSHINDIRECT** record defines a logical brush for graphics operations.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_CREATEBRUSHINDIRECT. This value is 0x00000027.

Size (4 bytes): An unsigned integer that specifies the size in bytes, of this record. This value is 0x00000018.

ihBrush (4 bytes): An unsigned integer that specifies the index of the logical brush object in the EMF object table (section 3.1.1.1). This index is used to refer to the object, so it can be reused or modified.

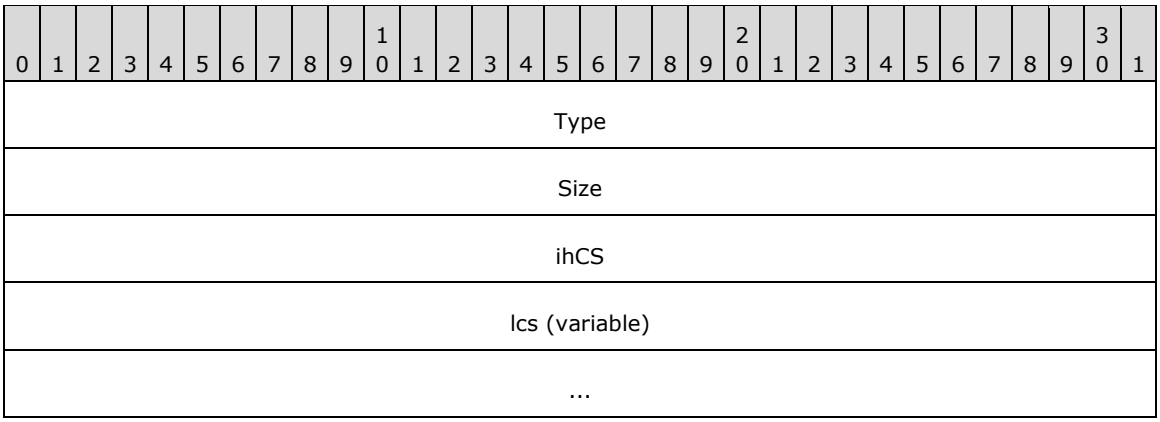
LogBrush (12 bytes): A LogBrushEx object (section 2.2.12) that specifies the style, color, and pattern of the logical brush. The **BrushStyle** field in this object MUST be BS_SOLID, BS_HATCHED, or BS_NULL.

See section 2.3.7 for more 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.<72>

Fields not specified in this section are specified in section 2.3.7.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CREATECOLORSPACE**. This value is 0x00000063.

ihCS (4 bytes): An 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 LogColorSpace object ([MS-WMF] section 2.2.2.11), which can specify the name of a color profile in ASCII characters.

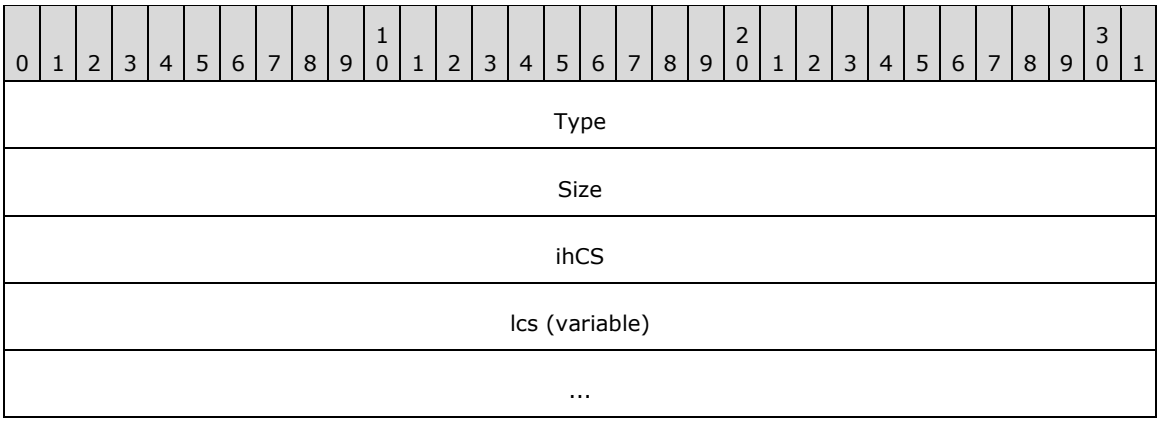
The logical color space object defined by this record can be selected into the playback device context by an EMR_SETCOLORSPACE record (section 2.3.8.7), which defines the logical color space to use in subsequent graphics operations.

See section 2.3.7 for more 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.<73>

Fields not specified in this section are specified in section 2.3.7.



dwFlags
cbData
Data (variable, optional)
...

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CREATECOLORSPACEW**. This value is 0x0000007A.

ihCS (4 bytes): An 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 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): An unsigned integer that provides information about the data in this record.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
0																															C

C (1 bit): If set, the **Data** field contains color profile data.

cbData (4 bytes): An unsigned integer that specifies the size in bytes, of the **Data** field.

Data (variable, optional): An 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 spaceto use in subsequent graphics operations.

See section 2.3.7 for more 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.

Fields not specified in this section are specified in section 2.3.7.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
ihBrush																															
Usage																															

offBmi
cbBmi
offBits
cbBits
BitmapBuffer (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_CREATEDIBPATTERNBRUSHPT. This value is 0x0000005E.

ihBrush (4 bytes): An 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): An unsigned integer that specifies how to interpret values in the color table in the DIB header. This value is in the DIBColors enumeration (section 2.1.9).

offBmi (4 bytes): An unsigned integer that specifies the offset from the start of this record to the DIB header.

cbBmi (4 bytes): An unsigned integer that specifies the size of the DIB header.

offBits (4 bytes): An unsigned integer that specifies the offset from the start of this record to the DIB bits.

cbBits (4 bytes): An unsigned integer that specifies the size of the DIB bits.

BitmapBuffer (variable): A buffer containing a packed DIB in the form of a DeviceIndependentBitmap object ([MS-WMF] section 2.2.2.9). It is not required to be contiguous with the fixed portion of this record.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
UndefinedSpace (variable, optional)																															
...																															
BmiSrc (variable)																															
...																															
BitsSrc (variable)																															
...																															

UndefinedSpace (variable, optional): An array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a DeviceIndependentBitmap object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a 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 more 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.

Fields not specified in this section are specified in section 2.3.7.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
ihBrush																															
Usage																															
offBmi																															
cbBmi																															
offBits																															
cbBits																															
BitmapBuffer (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CREATEMONOBRUSH**. This value is 0x0000005D.

ihBrush (4 bytes): An 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): An unsigned integer that specifies how to interpret values in the color table in the DIB header. This value is in the DIBColors enumeration (section 2.1.9).

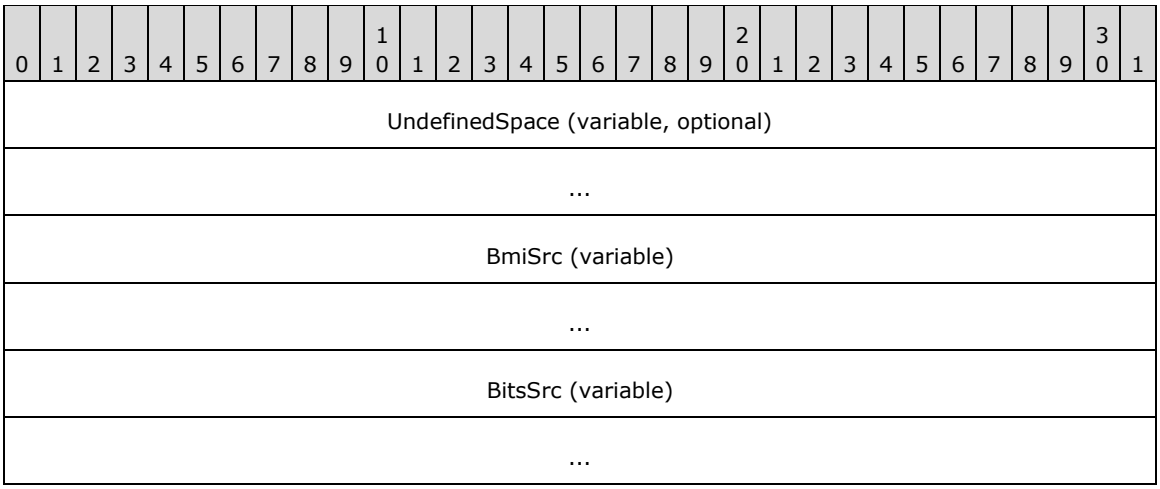
offBmi (4 bytes): An unsigned integer that specifies the offset from the start of this record to the DIB header.

cbBmi (4 bytes): An unsigned integer that specifies the size of the DIB header.

offBits (4 bytes): An unsigned integer that specifies the offset from the start of this record to the DIB bits.

cbBits (4 bytes): An unsigned integer that specifies the size of the DIB bits.

BitmapBuffer (variable): A buffer containing a packed DIB in the form of a monochrome DeviceIndependentBitmap object ([MS-WMF] section 2.2.2.9). It is not required to be contiguous with the fixed portion of this record.



UndefinedSpace (variable, optional): An array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a DeviceIndependentBitmap object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a DeviceIndependentBitmap object.

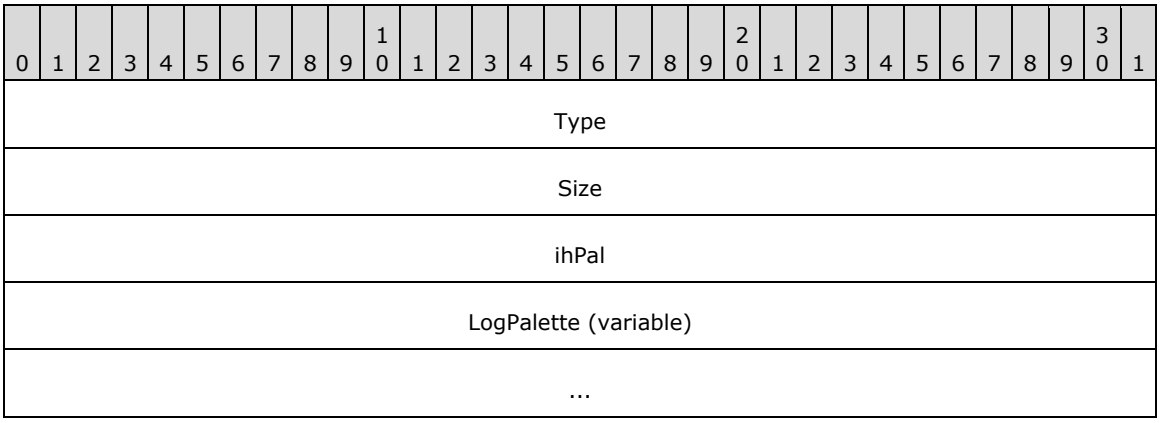
The monochrome pattern brush object defined by this record can be selected into the playback device context by an EMR_SELECTOBJECT record (section 2.3.8.5), which specifies the pattern brush to use in subsequent graphics operations.

See section 2.3.7 for more object creation record types.

2.3.7.6 EMR_CREATEPALETTE Record

The EMR_CREATEPALETTE record defines a logical palette for graphics operations.

Fields not specified in this section are specified in section 2.3.7.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CREATEPALETTE**. This value is 0x00000031.

ihPal (4 bytes): An 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.

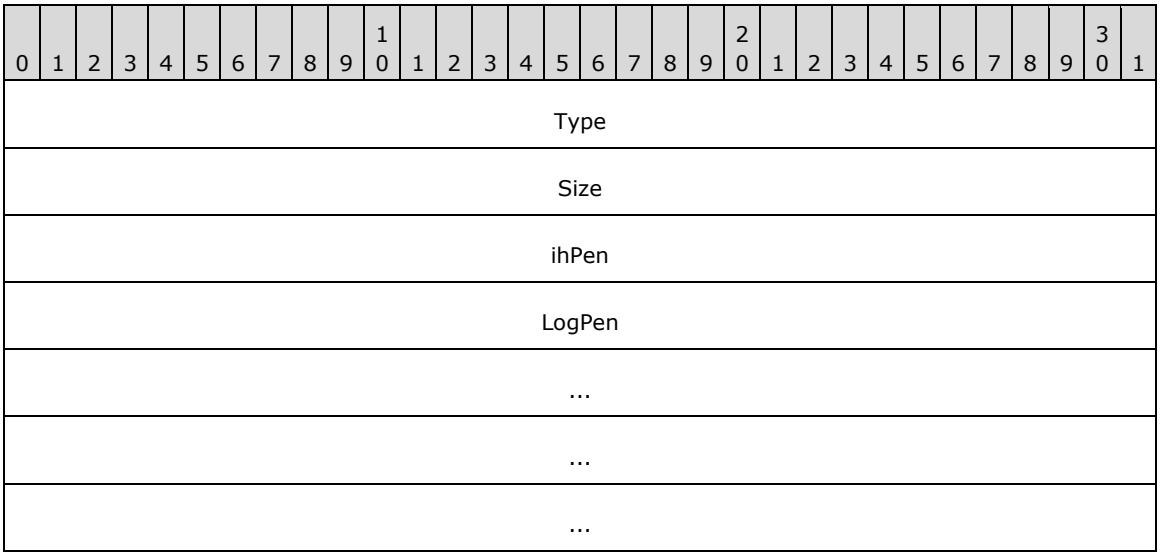
LogPalette (variable): A LogPalette object (section 2.2.17). The **Version** field of this object MUST be set to 0x0300. If the **NumberOfEntries** value in this object is zero, processing of this record MUST fail.

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 more object creation record types.

2.3.7.7 EMR_CREATEPEN Record

The EMR_CREATEPEN record defines a logical pen for graphics operations.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_CREATEPEN**. This value is 0x00000026.

Size (4 bytes): An unsigned integer that specifies the size in bytes, of this record. This value is 0x0000001C.

ihPen (4 bytes): An 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.

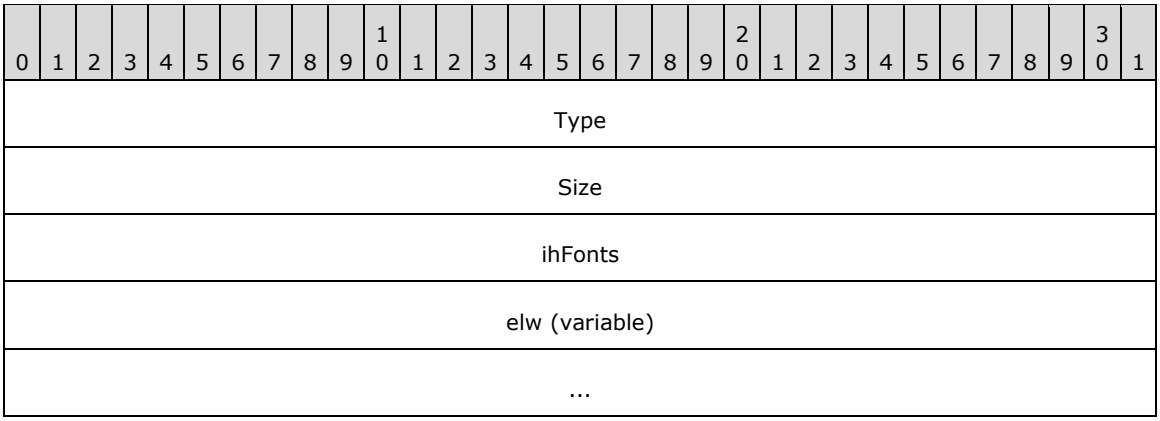
The logical pen object defined by this record can be selected into the playback device context by an EMR_SELECTOBJECT record (section 2.3.8.5), which specifies the logical pen to use in subsequent graphics operations.

See section 2.3.7 for more object creation record types.

2.3.7.8 EMR_EXTCREATEFONTINDIRECTW Record

The **EMR_EXTCREATEFONTINDIRECTW** record defines a logical font for graphics operations.

Fields not specified in this section are specified in section 2.3.7.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_EXTCREATEFONTINDIRECTW**. This value is 0x00000052.

ihFonts (4 bytes): An 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 (section 2.2.13) MAY<74> be present instead. 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 more 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.

Fields not specified in this section are specified in section 2.3.7.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
ihPen																															
offBmi																															
cbBmi																															
offBits																															
cbBits																															
elp (variable)																															
...																															
BitmapBuffer (variable, optional)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_EXTCREATEPEN**. This value is 0x0000005F.

ihPen (4 bytes): An 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): An unsigned integer that specifies the offset from the start of this record to the DIB header if the record contains a DIB.

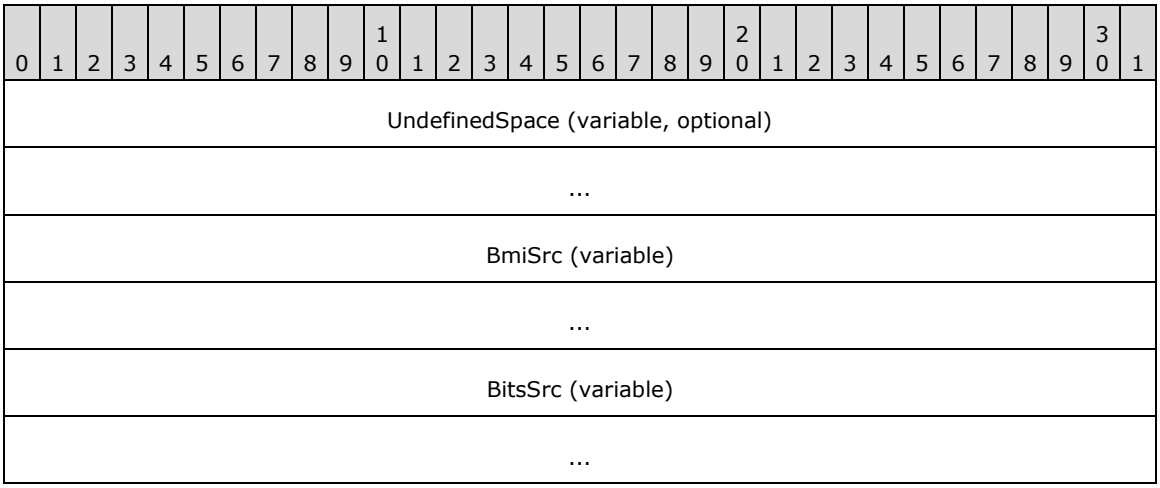
cbBmi (4 bytes): An unsigned integer that specifies the size of the DIB header if the record contains a DIB.

offBits (4 bytes): An 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): An 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, optional): An array of bytes containing a packed DIB in the form of a DeviceIndependentBitmap object ([MS-WMF] section 2.2.2.9). It is not required to be contiguous with the fixed portion of this record.



UndefinedSpace (variable, optional): An array of bytes that MUST be ignored.

BmiSrc (variable): The DIB header, which is the **DibHeaderInfo** field of a DeviceIndependentBitmap object.

BitsSrc (variable): The DIB bits, which is the **aData** field of a 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 more object creation record types.

2.3.8 Object Manipulation Record Types

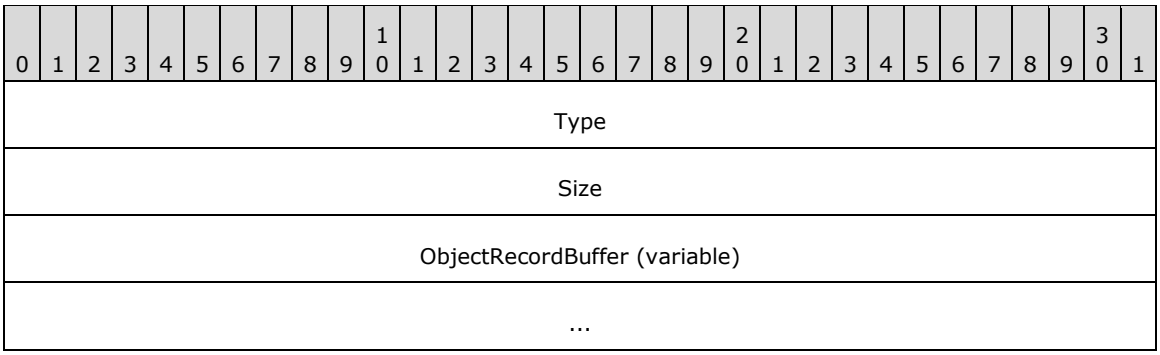
The **Object Manipulation** record types manage and modify graphics objects.

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 object (section 2.2.17) by using WCS values.

Name	Section	Description
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 and selects it into the playback device context
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.

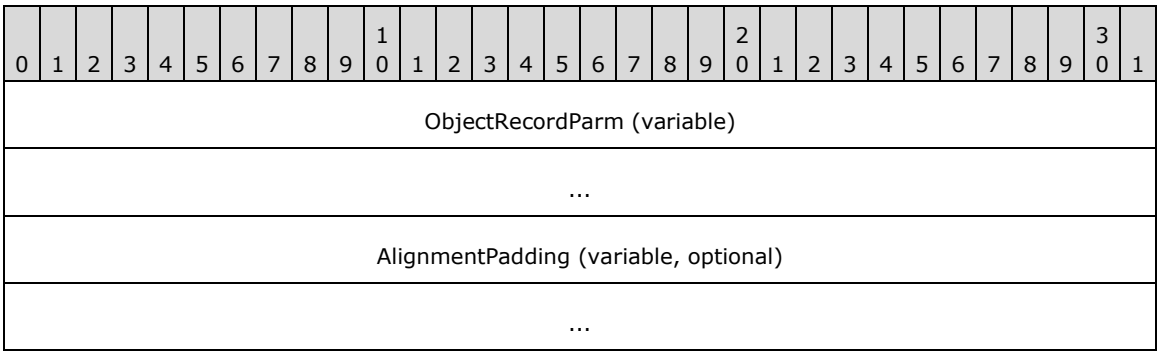


Type (4 bytes): An unsigned integer that defines the type of record. The object manipulation record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_SELECTOBJECT	0x00000025
EMR_DELETEOBJECT	0x00000028
EMR_SELECTPALETTE	0x00000030
EMR_SETPALETTEENTRIES	0x00000032
EMR_RESIZEPALETTE	0x00000033
EMR_SETCOLORSPACE	0x00000064
EMR_DELETECOLORSPACE	0x00000065
EMR_COLORCORRECTPALETTE	0x0000006F

Size (4 bytes): An 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.



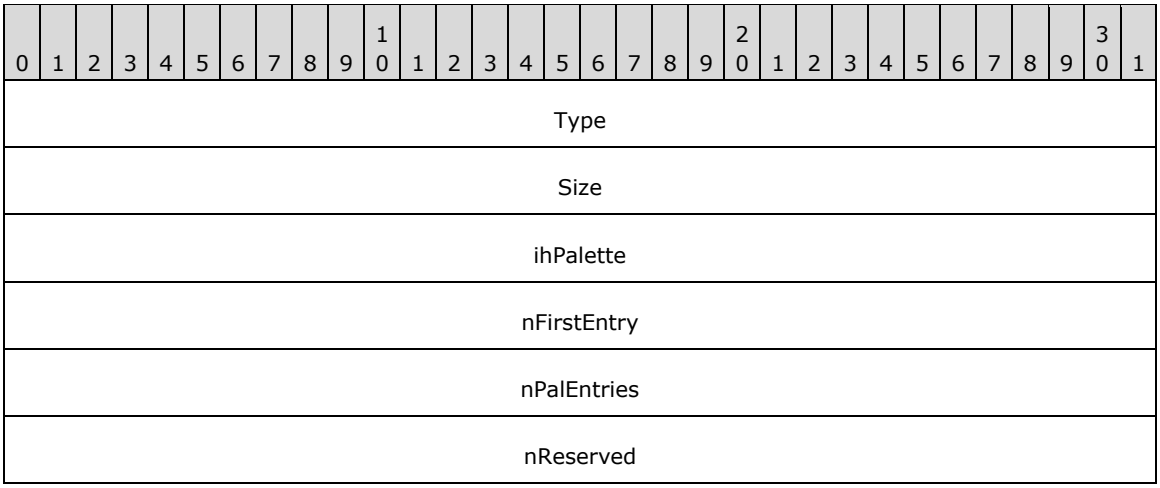
ObjectRecordParm (variable): An array of bytes that contains the parameters for the object manipulation record.

AlignmentPadding (variable, optional): An 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 more EMF record types.

2.3.8.1 EMR_COLORCORRECTPALETTE Record

The **EMR_COLORCORRECTPALETTE** record specifies the correction of entries of a logical palette object using WCS.<75>



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_COLORCORRECTPALETTE**. This value is 0x0000006F.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x00000018.

ihPalette (4 bytes): An unsigned integer that specifies the index of a logical palette object (section 2.2.17) in the **EMF object table** (section 3.1.1.1).

nFirstEntry (4 bytes): An unsigned integer that specifies the index of the first entry to correct.

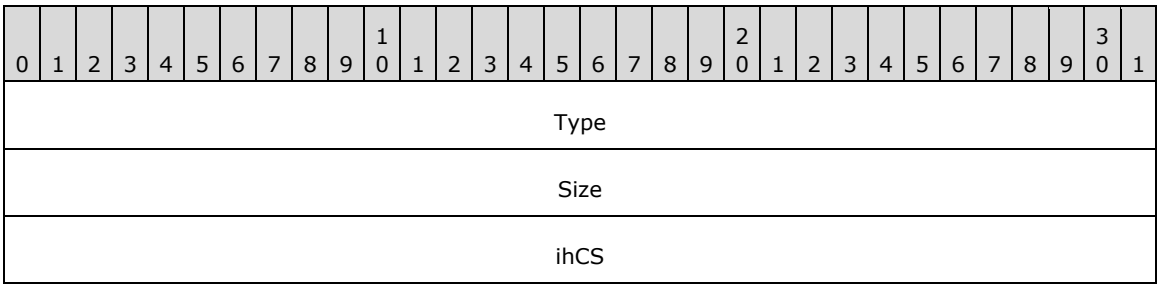
nPalEntries (4 bytes): An unsigned integer that specifies the number of palette entries to correct.

nReserved (4 bytes): An unsigned integer that is undefined and unused.

See section 2.3.8 for more object manipulation record types.

2.3.8.2 EMR_DELETECOLORSPACE Record

The **EMR_DELETECOLORSPACE** record deletes a logical color space object.<76>



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_DELETECOLORSPACE**. This value is 0x00000065.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ihCS (4 bytes): An unsigned integer that specifies the index of a logical color space object in the EMF object table (section 3.1.1.1).

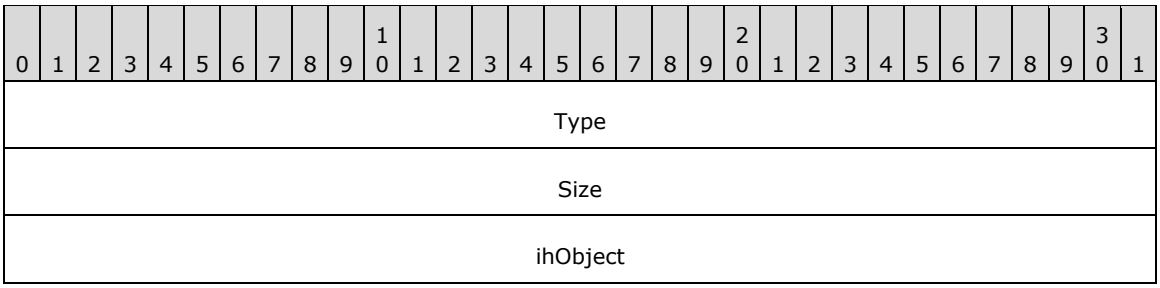
The color space is specified by either a LogColorSpace or LogColorSpaceW object ([MS-WMF] sections 2.2.2.11 and 2.2.2.12, respectively). If the deleted color space is currently selected into the playback device context, the default object MUST be restored.

An EMR_DELETEOBJECT record (section 2.3.8.3) SHOULD be used instead of this record to delete a logical color space object.

See section 2.3.8 for more 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).



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_DELETEOBJECT**. This value is 0x00000028.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ihObject (4 bytes): An unsigned integer that specifies the index of a graphics object in the EMF object table.

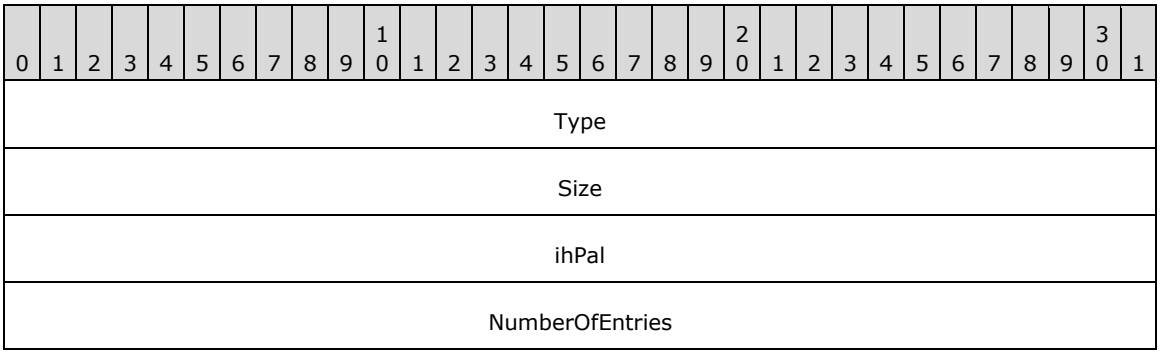
This value MUST NOT be 0, which is a reserved index that refers to the EMF metafile itself; and it MUST NOT be the index of a stock object, which cannot be deleted. Stock object indexes are specified in the StockObject (section 2.1.31) enumeration.

The object specified by this record MUST be deleted from the EMF object table. If the deleted object is currently selected into the playback device context, the default object MUST be restored.

See section 2.3.8 for more 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).



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_RESIZEPALETTE**. This value is 0x00000033.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ihPal (4 bytes): An unsigned integer that specifies the index of the palette object in the EMF object table (section 3.1.1.1).

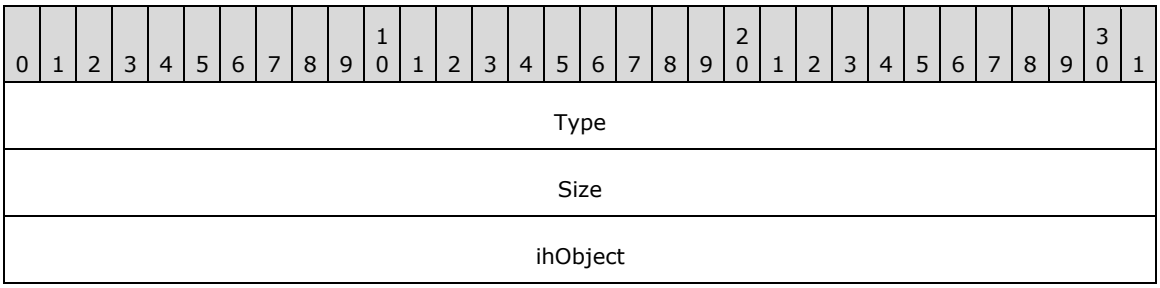
NumberOfEntries (4 bytes): An unsigned integer that specifies the number of entries in the palette after resizing. The value MUST be <= 0x00000400 and > 0x00000000.<77>

The new size of the LogPalette object MUST be reflected in the **NumberOfEntries** field in that structure.

See section 2.3.8 for more object manipulation record types.

2.3.8.5 EMR_SELECTOBJECT Record

The **EMR_SELECTOBJECT** record selects a graphics object into the playback device context.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SELECTOBJECT**. This value is 0x00000025.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ihObject (4 bytes): An unsigned integer that specifies either the index of a graphics object in the EMF object table (section 3.1.1.1) or the index of a stock object in the StockObject enumeration (section 2.1.31).

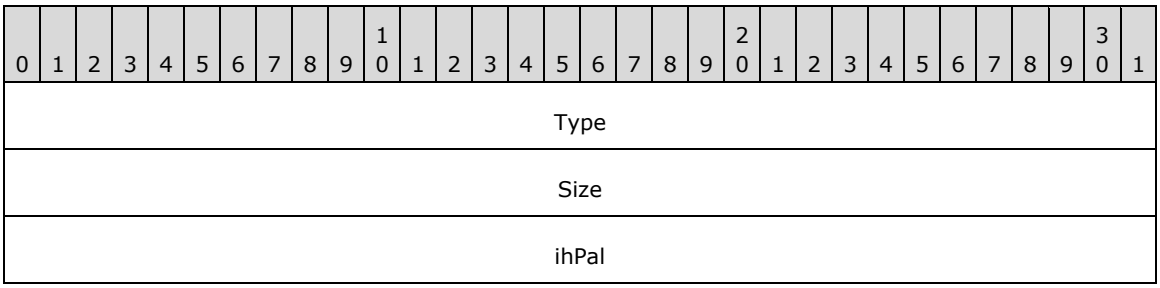
The object index MUST NOT be zero, which is reserved and refers to the EMF metafile itself.

The object specified by this record MUST be used in subsequent EMF drawing operations, until another EMR_SELECTOBJECT record changes the object of that type or the object is deleted.

See section 2.3.8 for more object manipulation record types.

2.3.8.6 EMR_SELECTPALETTE Record

The **EMR_SELECTPALETTE** record selects a logical palette into the playback device context.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SELECTPALETTE. This value is 0x00000030.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

ihPal (4 bytes): An unsigned integer that specifies either the index of a LogPalette object (section 2.2.17) in the EMF object table (section 3.1.1.1) or the value DEFAULT_PALETTE from the StockObject enumeration (section 2.1.31), which is the index of a stock palette.

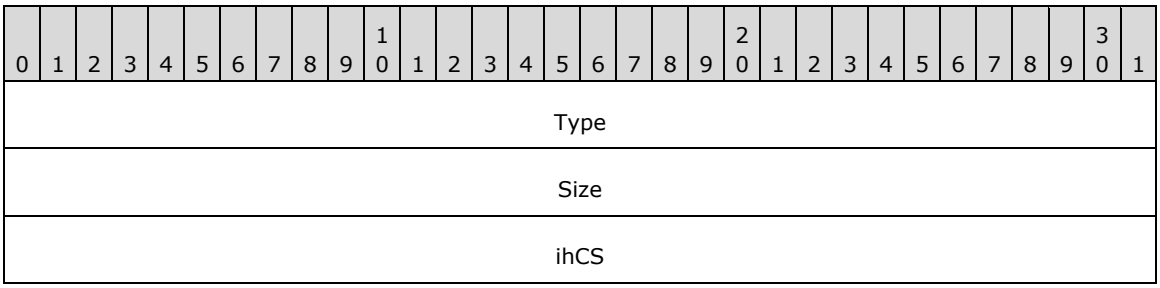
The object index MUST NOT be zero, which is reserved and refers to the EMF metafile itself.

The palette specified by this record MUST be used in subsequent EMF drawing operations, until another EMR_SELECTPALETTE record changes the object or the object is deleted.

See section 2.3.8 for more object manipulation record types.

2.3.8.7 EMR_SETCOLORSPACE Record

The **EMR_SETCOLORSPACE** record selects a logical color space into the playback device context.<78>



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETCOLORSPACE**. This value is 0x00000064.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

ihCS (4 bytes): An unsigned integer that specifies the index of a logical color space object in the EMF object table (section 3.1.1.1).

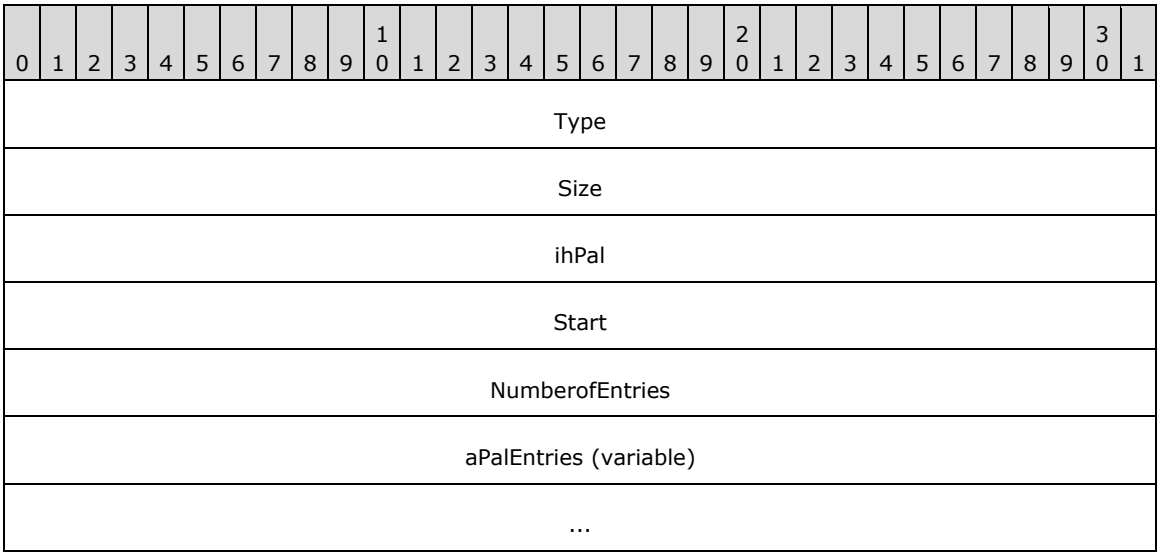
This object is either a LogColorSpace or LogColorSpaceW object ([MS-WMF] sections 2.2.2.11 and 2.2.2.12, respectively).

The color space specified by this record MUST be used in subsequent EMF drawing operations, until another EMR_SETCOLORSPACE record changes the object or the object is deleted.

See section 2.3.8 for more 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 logical palette.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETPALETTEENTRIES. This value is 0x00000032.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ihPal (4 bytes): An unsigned integer that specifies an index of a LogPalette object (section 2.2.17) in the EMF object table (section 3.1.1.1).

Start (4 bytes): An unsigned integer that specifies the index in the palette of the first entry to set.

NumberofEntries (4 bytes): An unsigned integer that specifies the number of entries in the **aPalEntries** array.

aPalEntries (variable): An array of LogPaletteEntry objects (section 2.2.18) that specify the palette data.

See section 2.3.8 for more object manipulation record types.

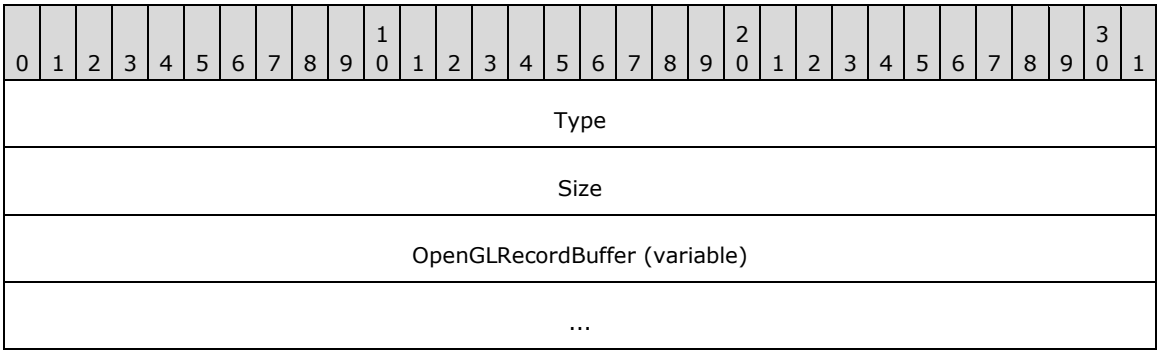
2.3.9 OpenGL Record Types

The **OpenGL** record types specify OpenGL functions [OPENGL].

The following are EMF OpenGL record types.<79>

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.

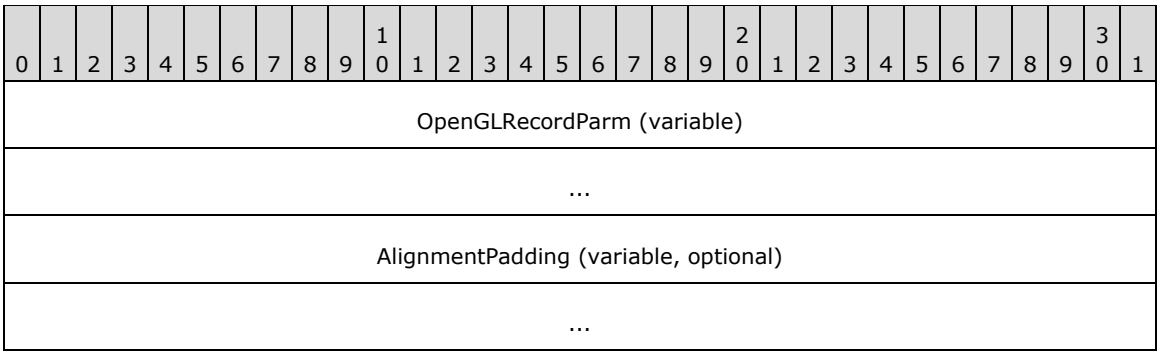


Type (4 bytes): An unsigned integer that defines the type of record. The OpenGL record types are listed in the following table. See the preceding table for descriptions of these record types.

Name	Value
EMR_GLSRECORD	0x00000066
EMR_GLSBOUNDEDRECORD	0x00000067

Size (4 bytes): An 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.



OpenGLRecordParm (variable): An array of bytes that contains the parameters for the OpenGL record.

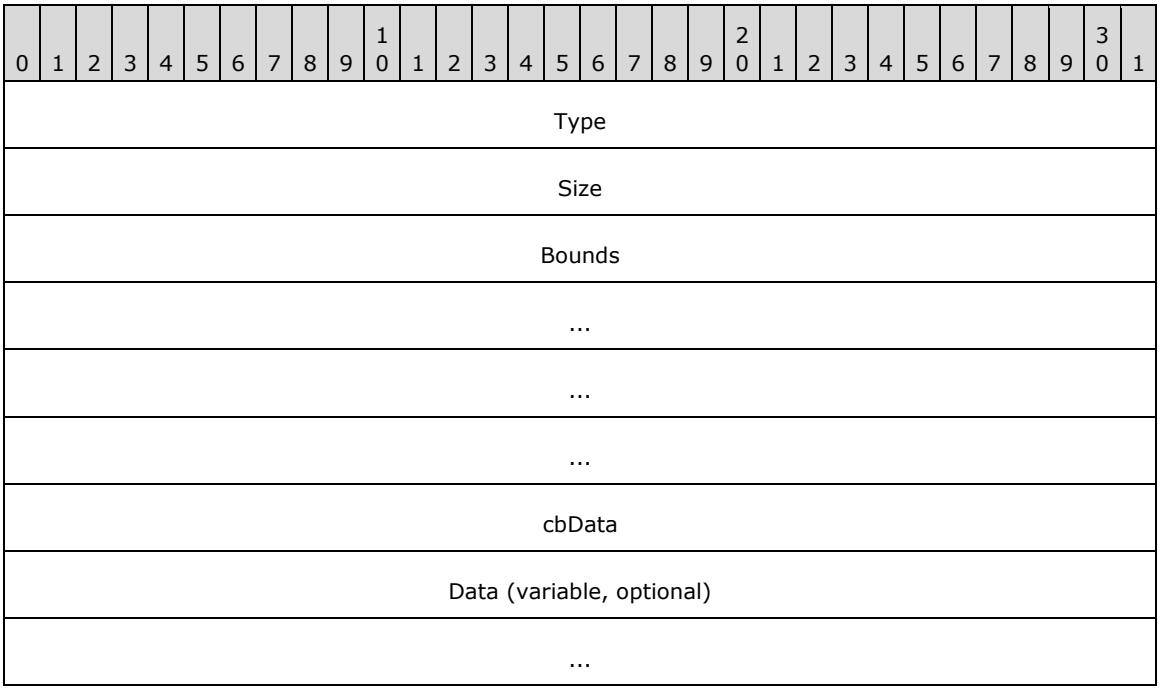
AlignmentPadding (variable, optional): An array of up to 3 bytes that pads the record so that its total size is a multiple of 4 bytes. This field MUST be ignored.

See section 2.3 for more EMF record types.

2.3.9.1 EMR_GLSBOUNDEDRECORD Record

The **EMR_GLSBOUNDEDRECORD** record specifies an OpenGL function with a bounding rectangle for output.

Fields not specified in this section are specified in section 2.3.1.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_GLSBOUNDEDRECORD. This value is 0x00000067.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that defines a bounding rectangle in logical units, for output produced by executing the OpenGL function.

cbData (4 bytes): An unsigned integer that specifies the size of the **Data** field in bytes. If this value is zero, no data is attached to this record.

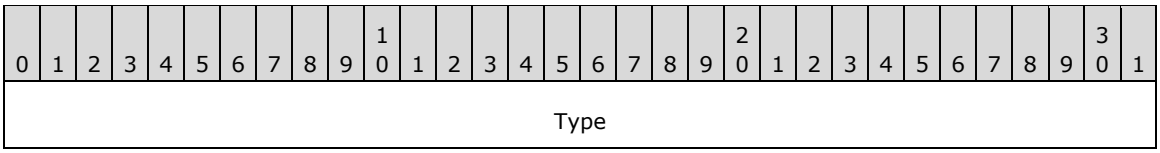
Data (variable, optional): An array of bytes that specifies data for the OpenGL function.

See section 2.3.9 for more OpenGL record types.

2.3.9.2 EMR_GLSRECORD Record

The **EMR_GLSRECORD** record specifies an OpenGL function.

Fields not specified in this section are specified in section 2.3.1.



Size
cbData
Data (variable, optional)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_GLSRECORD. This value is 0x00000066.

cbData (4 bytes): An 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, optional): An array of bytes that specifies data for the OpenGL function.

See section 2.3.9 for more OpenGL record types.

2.3.10 Path Bracket Record Types

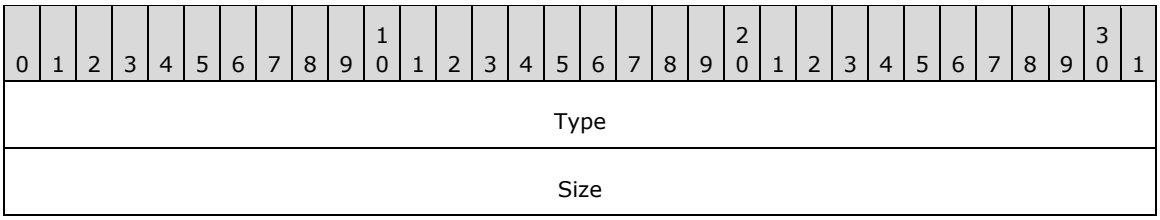
The **Path Bracket** record types are used to construct a path bracket, which defines the current path in the playback device context.

Note: None of the path bracket records specify parameters.

The following are the path bracket record types.

Name	Description
EMR_ABORTPATH	This record closes path bracket construction and discards the current path.
EMR_BEGINPATH	This record opens path bracket construction. Once path bracket construction is open, an application can begin specifying records to define the points that lie in the path. Path bracket construction MUST be closed by an EMR_ABORTPATH or EMR_ENDPATH record. When an application processes an EMR_BEGINPATH record, path bracket construction MUST NOT be open.
EMR_CLOSEFIGURE	This record closes the figure in path bracket construction. Processing the EMR_CLOSEFIGURE record closes the figure by drawing a line from the current drawing position to the first point of the figure, and then it connects the lines by using the current line join. If the figure is closed by processing an EMR_LINETO record (section 2.3.5.13) instead of this record, the current line cap is used to create the corner instead of the line join. The line parameters are specified by the **PenStyle** field in the current LogPen (section 2.2.19) and LogPenEx (section 2.2.20) objects. The EMR_CLOSEFIGURE record SHOULD be used only if there is an open figure in the path bracket. A figure in a path is open unless it is explicitly closed by processing this record. A figure can be open even if the current point is the same as the starting point. After processing the EMR_CLOSEFIGURE record, adding a line or curve to the path bracket starts a new figure.
EMR_ENDPATH	This record closes path bracket construction and selects the path into the playback device context.
EMR_FLATTENPATH	This record transforms each curve in the current path into a sequence of lines.
EMR_WIDENPATH	This record redefines the current path as the area that would be painted if its path were drawn using the current pen.

The generic structure of path bracket records is specified as follows.



Type (4 bytes): An 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): An unsigned integer that specifies the size in bytes of this record in the metafile. For path bracket records, this value is 0x00000008

See section 2.3 for more EMF record types.

2.3.11 State Record Types

The **State** record types specify graphics properties that define the playback device context during the processing of EMF metafile records. For more information about the state used for metafile playback, see the Abstract Data Model section 3.1.1.

Note: The EMR_REALIZEPALETTE and EMR_SAVEDC records do not specify parameters.

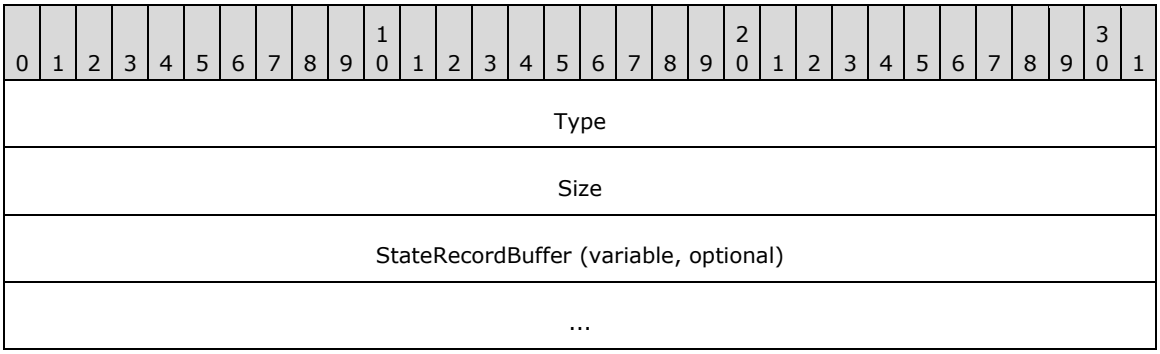
The following are EMF state record types.

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 LogFont (section 2.2.13).
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.

Name	Section	Description
		This 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 the playback device context in an array of states saved by preceding **EMR_SAVEDC** records if any. An **EMR_RESTOREDC** record is used to restore the state. This record specifies no parameters.
EMR_SCALEVIEWPORTEXTEX	2.3.11.7	Specifies the viewport by using the ratios formed by the specified multiplicands and divisors.
EMR_SCALEWINDOWEXTEX	2.3.11.8	Specifies the window 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.
EMR_SETICMPROFILEA	2.3.11.15	Specifies how to set a specified color profile as the output profile.
EMR_SETICMPROFILEW	2.3.11.16	Specifies how to set a specified color profile as the output profile.
EMR_SETLAYOUT	2.3.11.17	Specifies the layout of the playback device context.
EMR_SETLINKEDUFIS	2.3.11.18	Sets the **UniversalFontId** (section 2.2.27) of the linked fonts to use during character lookup.
EMR_SETMAPMODE	2.3.11.19	Specifies the mapping mode.
EMR_SETMAPPERFLAGS	2.3.11.20	Specifies 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.
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.

Name	Section	Description
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): An 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.

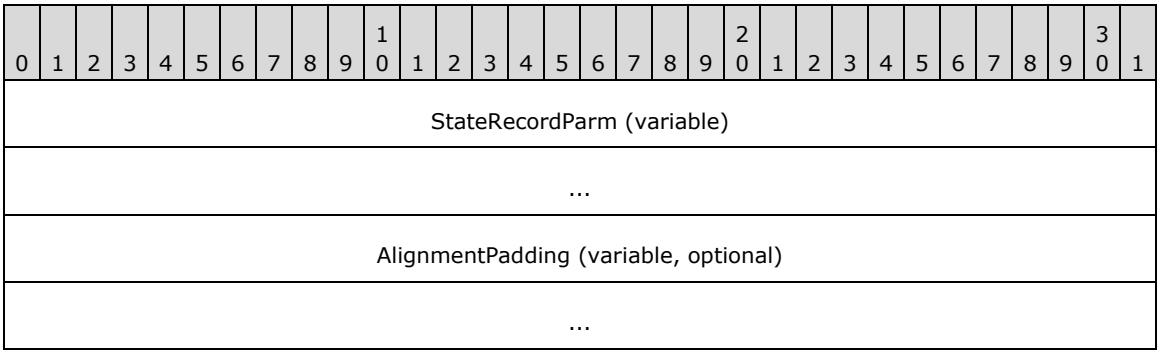
Name	Value
EMR_SETWINDOWORGEX	0x0000000A
EMR_SETVIEWPORTEXTEX	0x0000000B
EMR_SETVIEWPORTORGEX	0x0000000C
EMR_SETBRUSHORGEX	0x0000000D
EMR_SETCOLORADJUSTMENT	0x00000017
EMR_MOVETOEX	0x0000001B
EMR_SCALEVIEWPORTEXTEX	0x0000001F
EMR_SETWINDOWEXTEX	0x00000009
EMR_SETMAPPERFLAGS	0x00000010
EMR_SETMAPMODE	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

Name	Value
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
EMR_SETICMPROFILEA	0x00000070
EMR_SETICMPROFILEW	0x00000071
EMR_SETLAYOUT	0x00000073
EMR_SETLINKEDUFIS	0x00000077
EMR_SETTEXTJUSTIFICATION	0x00000078
EMR_COLORMATCHTOTARGETW	0x00000079

Size (4 bytes): An 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, optional): An array of bytes that contains the remainder of the state record.

The EMR_REALIZEPALETTE and EMR_SAVEDC records do not contain this field.



StateRecordParm (variable): An array of bytes that contains the parameters for the state record.

AlignmentPadding (variable, optional): An 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 more 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.<80>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
dwAction																															
dwFlags																															
cbName																															
cbData																															
Data (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_COLORMATCHTOTARGETW. This value is 0x00000079.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

dwAction (4 bytes): An unsigned integer that specifies a value from the ColorSpace enumeration (section 2.1.7).

dwFlags (4 bytes): An unsigned integer that specifies a value from the ColorMatchToTarget enumeration (section 2.1.6).

cbName (4 bytes): An unsigned integer that specifies the number of bytes in the Unicode UTF16-LE name of the target color profile.

cbData (4 bytes): An unsigned integer that specifies the size of the raw data of the target color profile in the **Data** field.

Data (variable): An array of size (**cbName** + **cbData**) bytes, which specifies the UTF16-LE name and raw data of the target color profile.

An EMR_COLORMATCHTOTARGETW record can be used to control whether to apply the current color transform to subsequent graphics operations. If the **dwAction** field value is **CS_ENABLE**, color mapping is enabled, and the current color transform SHOULD be applied. If **dwAction** is set to **CS_DISABLE**, the color transform SHOULD NOT be applied.

Before applying the current color transform, WCS SHOULD be enabled in the playback device context.<81>

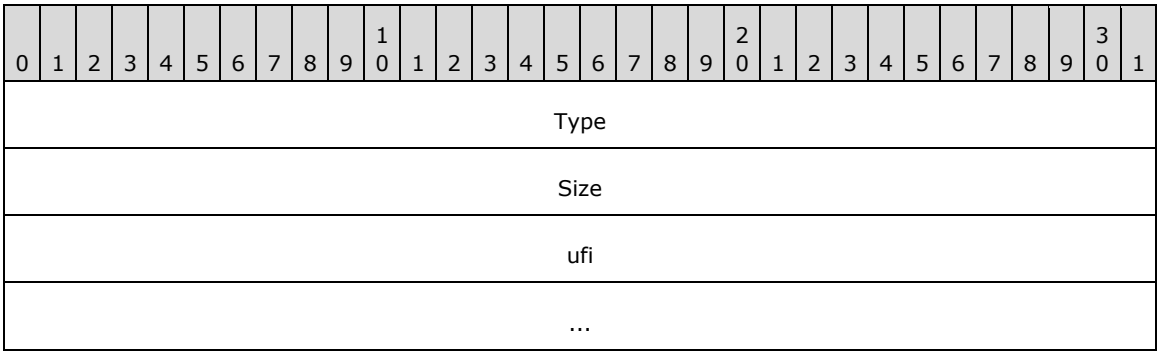
While color mapping to the target is enabled by a **dwAction** value of **CS_ENABLE**, changes to the color space or color gamut mapping are not applied. However, those changes MUST take effect when color mapping to the target is disabled.

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 more 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.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_FORCEUFIMAPPING. This value is 0x0000006D.

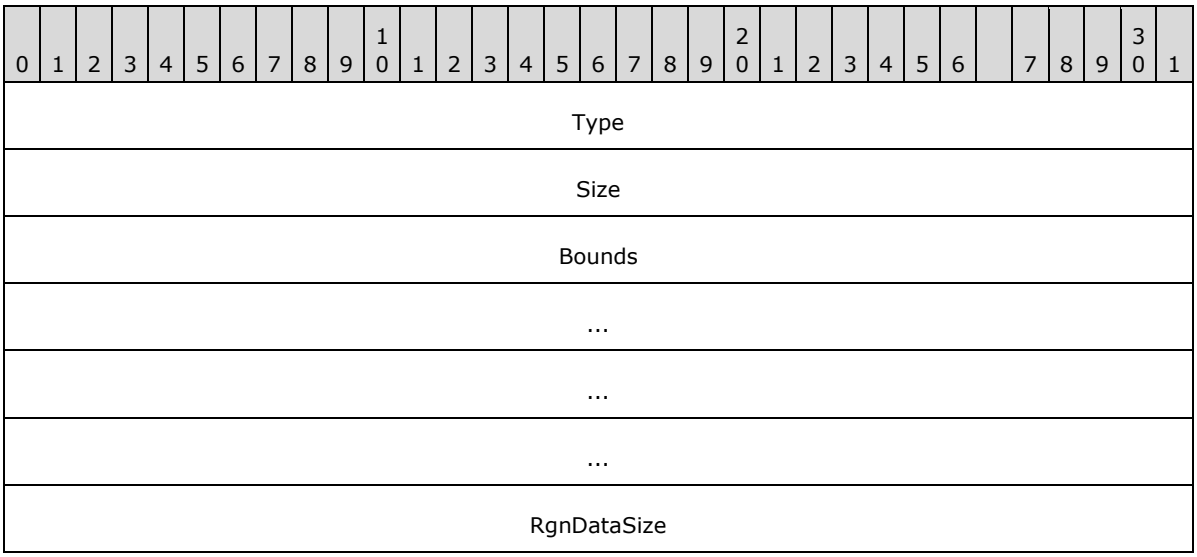
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

ufi (8 bytes): The font id to use, specified as a UniversalFontId (section 2.2.27).

See section 2.3.5 for more drawing record types.

2.3.11.3 EMR_INVERTRGN Record

The **EMR_INVERTRGN** record inverts the colors in the specified region. The current clipping regions used by this record are maintained in a **Regions** state element (section 3.1.1.2.1) in the playback device context (section 3.1).



RgnData (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_INVERTRGN. This value is 0x00000049.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Bounds (16 bytes): A RectL object ([MS-WMF] section 2.2.2.19) that specifies the destination bounding rectangle in logical coordinates. If the intersection of this rectangle with the current clipping region is empty, this record has no effect.

RgnDataSize (4 bytes): An unsigned integer that specifies the size of region data in bytes.

RgnData (variable): A **RgnDataSize** length array of bytes that specifies the output region in a **RegionData** object (section 2.2.24). The bounds specified by the **RegionDataHeader** field of this object MAY<82> be used as the bounding rectangle of the region when this record is processed.

If the output surface is monochrome, this record SHOULD convert white pixels to black and black pixels to white. For color output, the inversion is dependent on the type of technology used to generate the colors.

See section 2.3.11 for more state record types.

2.3.11.4 EMR_MOVETOEX Record

The EMR_MOVETOEX record specifies the coordinates of s new drawing position in logical units.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Offset																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_MOVETOEX**. This value is 0x0000001B.

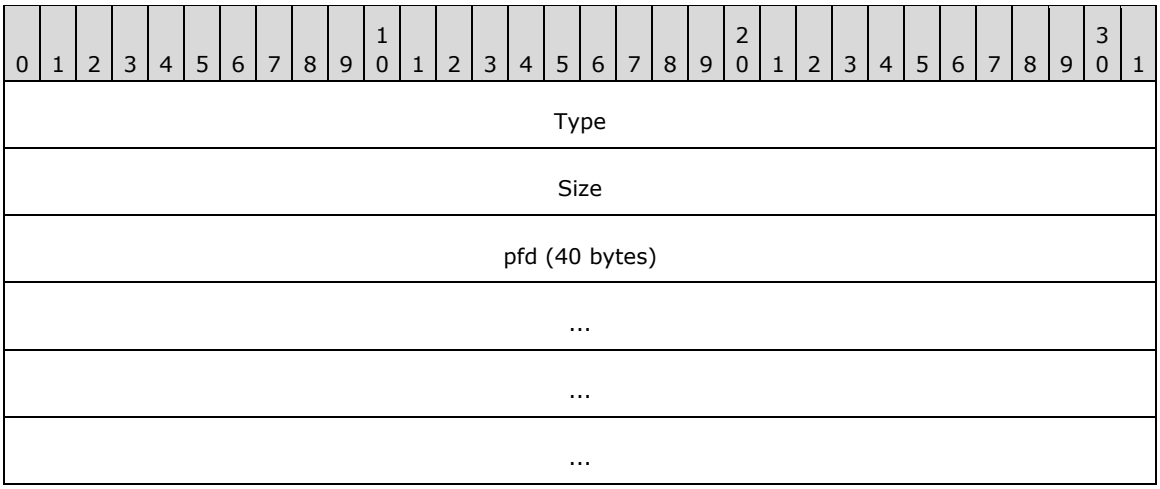
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Offset (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies coordinates of the new drawing position in logical units.

See section 2.3.11 for more state record types.

2.3.11.5 EMR_PIXELFORMAT Record

The EMR_PIXELFORMAT record specifies the pixel format to use for graphics operations.<83>



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_PIXELFORMAT**. This value is 0x00000068.

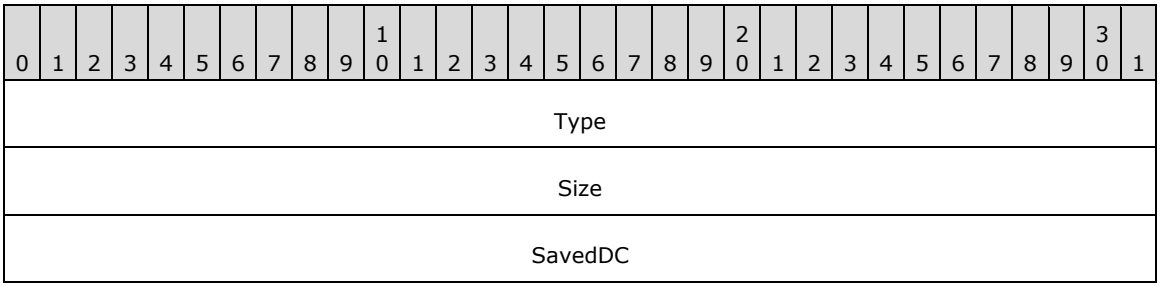
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

pfd (40 bytes): A PixelFormatDescriptor object (section 2.2.22) that specifies pixel format data.

See section 2.3.11 for more 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 a prior EMR_SAVEDC record (section 2.3.11).



Type (4 bytes): An unsigned integer that identifies the record type as **EMR_RESTOREDC**. This value is 0x00000022.

Size (4 bytes): An unsigned integer that specifies the size of the record in bytes. This value is 0x0000000C.

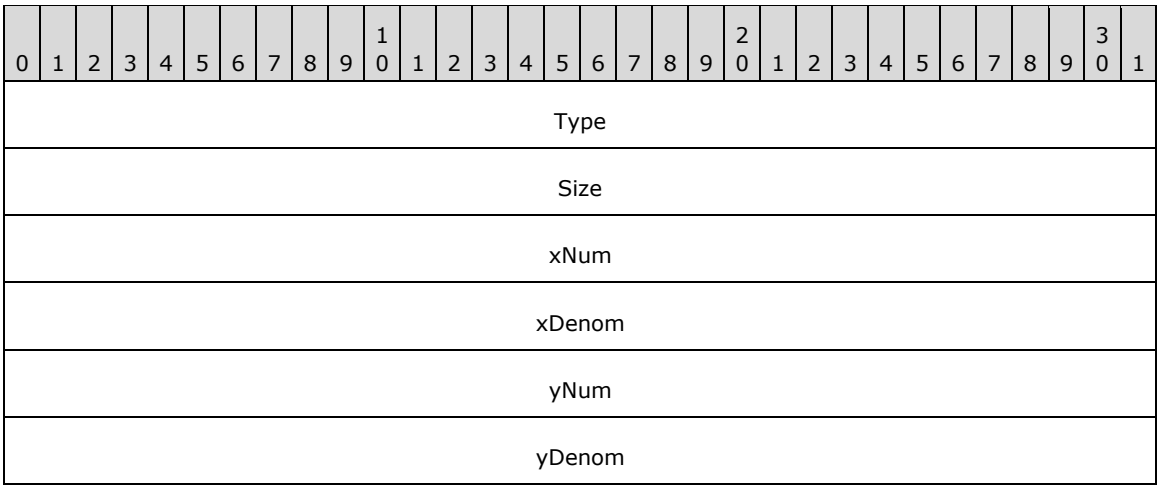
SavedDC (4 bytes): A signed integer that specifies the saved state to restore relative to the current state. This value MUST be negative; –1 represents the state that was most recently saved on the stack, –2 the one before that, etc.

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 more state record types.

2.3.11.7 EMR_SCALEVIEWPORTEXTEX Record

The **EMR_SCALEVIEWPORTEXTEX** record specifies the current viewport in the playback device context by using ratios formed by the specified multiplicands and divisors.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SCALEVIEWPORTEXTEX. This value is 0x0000001F.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

xNum (4 bytes): A signed integer that specifies the horizontal multiplicand. Cannot be zero.

xDenom (4 bytes): A signed integer that specifies the horizontal divisor. Cannot be zero.

yNum (4 bytes): A signed integer that specifies the vertical multiplicand. Cannot be zero.

yDenom (4 bytes): A signed integer that specifies the vertical divisor. Cannot be zero.

The extent MUST NOT be changed if the current mapping mode (section 2.1.21) is fixed scale. Only MM_ISOTROPIC and MM_ANISOTROPIC are not fixed scale.

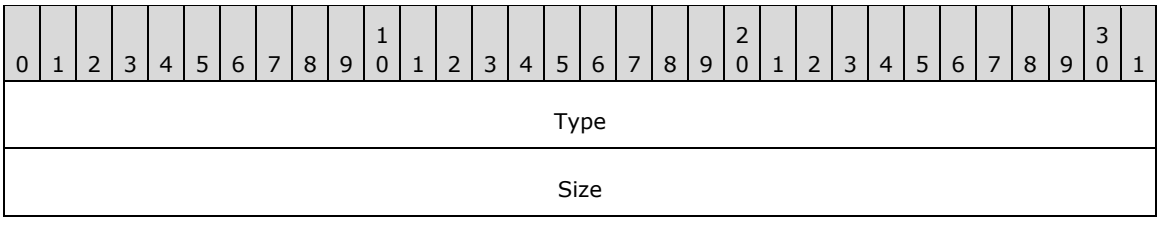
The new viewport extent is computed as follows.

```
xNewWE = (xOldWE * xNum) / xDenom
yNewWE = (yOldWE * yNum) / yDenom
```

See section 2.3.11 for more state record types.

2.3.11.8 EMR_SCALEWINDOWEXTEX Record

The **EMR_SCALEWINDOWEXTEX** record specifies the current window in the playback device context by using ratios formed by the specified multiplicands and divisors.



xNum
xDenom
yNum
yDenom

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SCALEWINDOWEXTEX. This value is 0x00000020.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

xNum (4 bytes): A signed integer that specifies the horizontal multiplicand. MUST NOT be zero.

xDenom (4 bytes): A signed integer that specifies the horizontal divisor. MUST NOT be zero.

yNum (4 bytes): A signed integer that specifies the vertical multiplicand. MUST NOT be zero.

yDenom (4 bytes): A signed integer that specifies the vertical divisor. MUST NOT be zero.

The extent MUST NOT be changed if the current mapping mode (section 2.1.21) is fixed scale. Only MM_ISOTROPIC and MM_ANISOTROPIC are not fixed scale.

The new window extent is computed as follows.

```
xNewWE = (xOldWE * xNum) / xDenom
yNewWE = (yOldWE * yNum) / yDenom
```

See section 2.3.11 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
ArcDirection																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETARCDIRECTION. This value is 0x00000039.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

ArcDirection (4 bytes): An unsigned integer that specifies the arc direction. This value is in the ArcDirection enumeration (section 2.1.2). The default direction is counterclockwise.

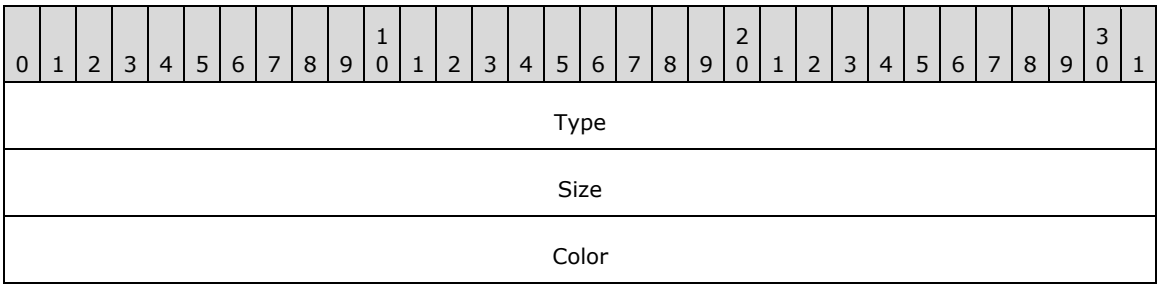
The arc direction affects the direction in which the following records draw:

- 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 more state record types.

2.3.11.10 EMR_SETBKCOLOR Record

The **EMR_SETBKCOLOR** record specifies the background color for text output.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETBKCOLOR. This value is 0x00000019.

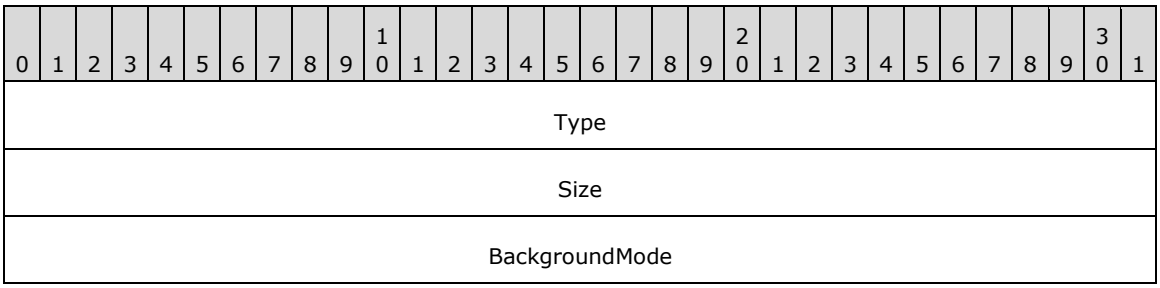
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Color (4 bytes): A ColorRef object ([MS-WMF] section 2.2.2.8), which specifies the background color value.

See section 2.3.11 for more state record types.

2.3.11.11 EMR_SETBKMODE Record

The **EMR_SETBKMODE** record specifies the background mix mode to use with text, hatched brushes, and pens that are not solid lines.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETBKMODE. This value is 0x00000012.

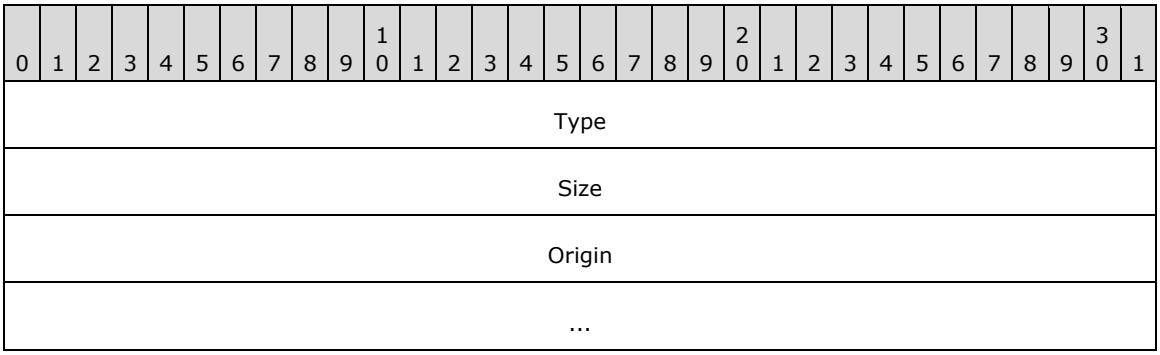
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

BackgroundMode (4 bytes): An unsigned integer that specifies the background mode, from the BackgroundMode enumeration (section 2.1.4).

See section 2.3.11 for more state record types.

2.3.11.12 EMR_SETBRUSHORGEX Record

The **EMR_SETBRUSHORGEX** record specifies the origin of the current brush.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETBRUSHORGEX. This value is 0x0000000D.

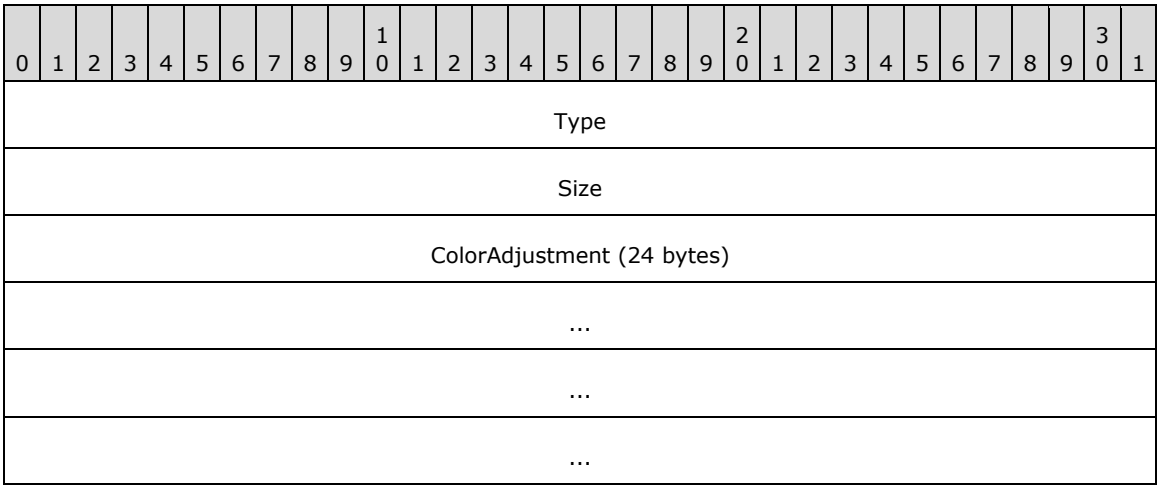
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Origin (8 bytes): A PointL object ([MS-WMF] section 2.2.2.15), which specifies the horizontal and vertical origin of the current brush in logical units.

See section 2.3.11 for more state record types.

2.3.11.13 EMR_SETCOLORADJUSTMENT Record

The **EMR_SETCOLORADJUSTMENT** record specifies color adjustment properties in the playback device context.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETCOLORADJUSTMENT**. This value is 0x00000017.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x00000020.

ColorAdjustment (24 bytes): A ColorAdjustment object (section 2.2.2) that specifies color adjustment values.

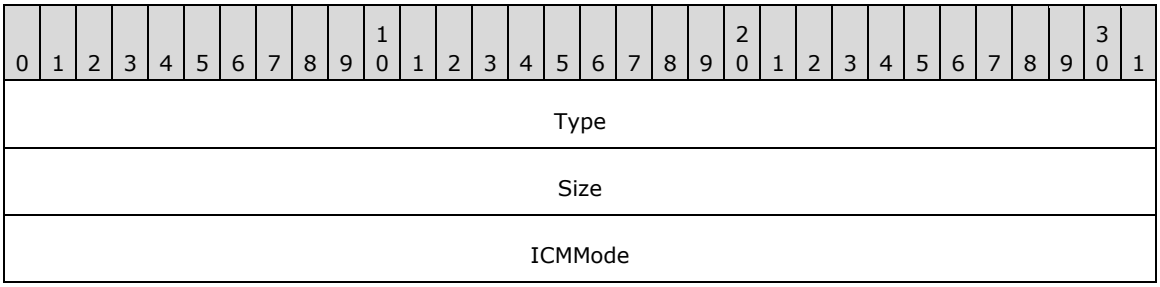
Color adjustment values are used to adjust the input color of the source bitmap for graphics operations performed by EMR_STRETCHBLT and EMR_STRETCHDIBITS records when **STRETCH_HALFTONE** mode is set from the StretchMode enumeration (section 2.1.32).

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 more 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.<84>



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETICMMODE**. This value is 0x00000062.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

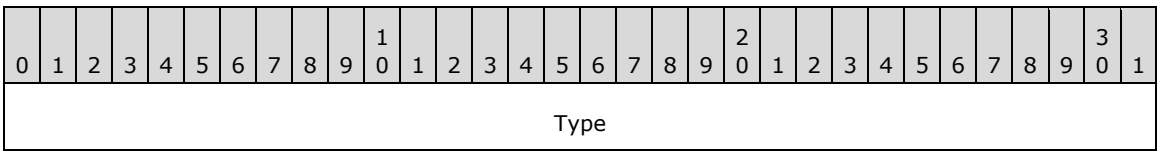
ICMMode (4 bytes): An unsigned integer that specifies whether to enable or disable ICM, from the ICMMode enumeration (section 2.1.18).

When ICM mode is enabled in the playback device context, colors specified in EMF records SHOULD be color matched, whereas the default color profile 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 more 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.<85>



Size
dwFlags
cbName
cbData
Data (variable)
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETICMPROFILEA. This value is 0x00000070.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

dwFlags (4 bytes): An unsigned integer that contains color profile flags.

cbName (4 bytes): An unsigned integer that specifies the number of bytes in the ASCII name of the desired color profile.

cbData (4 bytes): An 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 more 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.<86>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
dwFlags																															
cbName																															
cbData																															
Data (variable)																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETICMPROFILEW**. This value is 0x00000071.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

dwFlags (4 bytes): An unsigned integer that contains color profile flags.

cbName (4 bytes): An unsigned integer that specifies the number of bytes in the Unicode UTF16-LE name of the desired color profile.

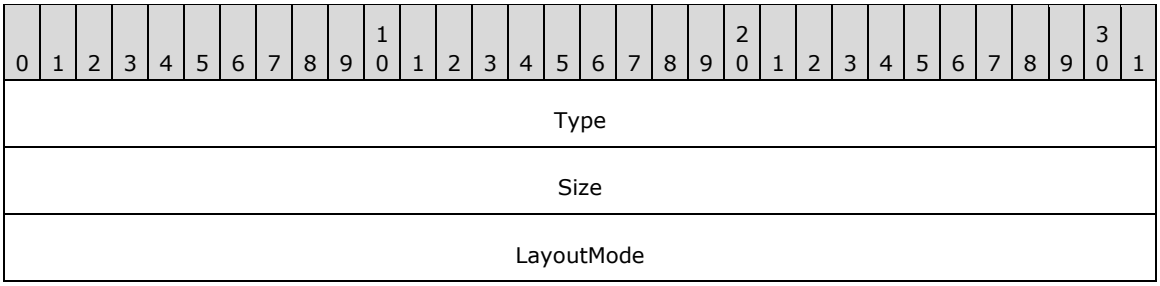
cbData (4 bytes): An unsigned integer that specifies the size of color profile data, if attached.

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 more state record types.

2.3.11.17 EMR_SETLAYOUT Record

The **EMR_SETLAYOUT** record specifies the order in which text and graphics are drawn.<87>



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETLAYOUT. This value is 0x00000073.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

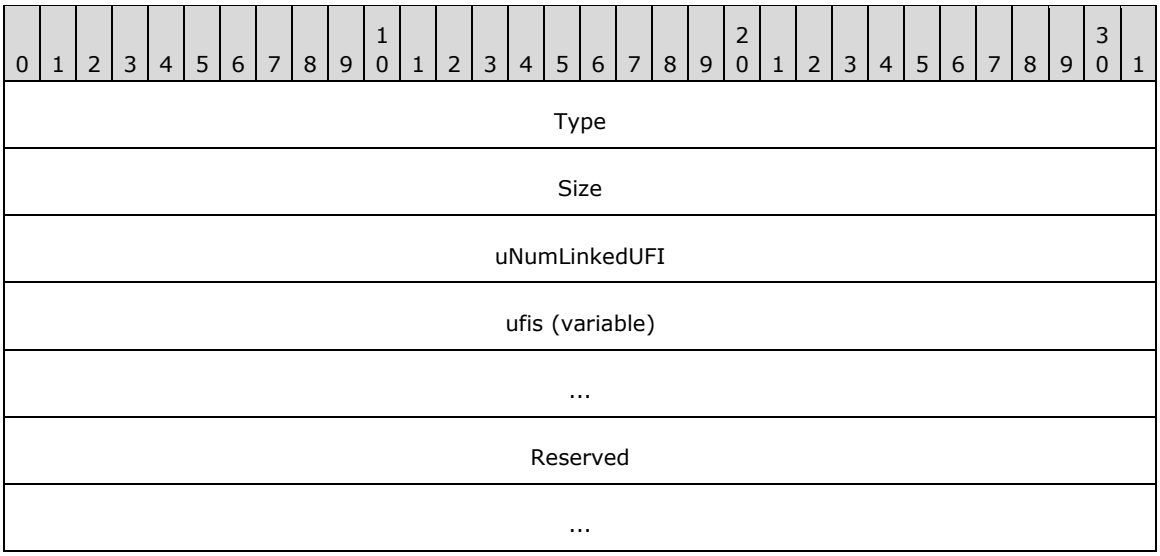
LayoutMode (4 bytes): An unsigned integer that specifies the layout mode as follows:

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 records (section 2.3.1) when the layout mode is right-to-left.

See section 2.3.11 for more state record types.

2.3.11.18 EMR_SETLINKEDUFIS Record

The **EMR_SETLINKEDUFIS** record sets the **UniversalFontIds** (section 2.2.27) of linked fonts to use during character lookup.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETLINKEDUFIS**. This value is 0x00000077.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

uNumLinkedUFI (4 bytes): An unsigned integer specifying the number of UFIs to follow.

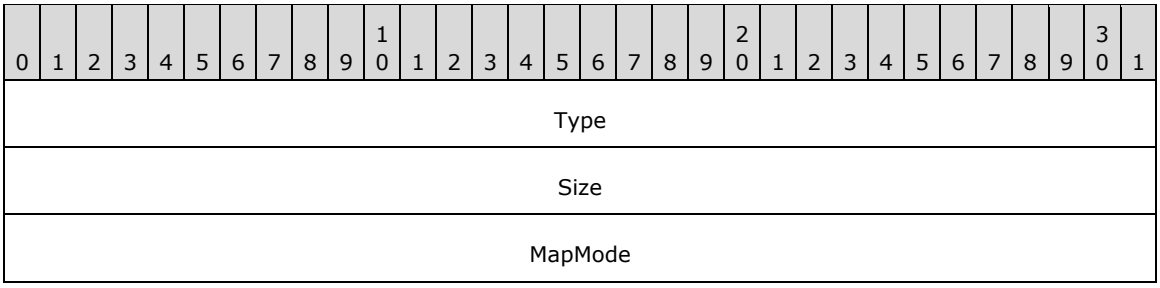
ufis (variable): An array of **uNumLinkedUFI** elements of type UniversalFontId (section 2.2.27), which specifies the identifiers of the linked fonts.

Reserved (8 bytes): This field is reserved and MUST be ignored.

See section 2.3.11 for more state record types.

2.3.11.19 EMR_SETMAPMODE Record

The **EMR_SETMAPMODE** record specifies the current mapping mode, which 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETMAPMODE**. This value is 0x00000011.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

MapMode (4 bytes): An unsigned integer from 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.

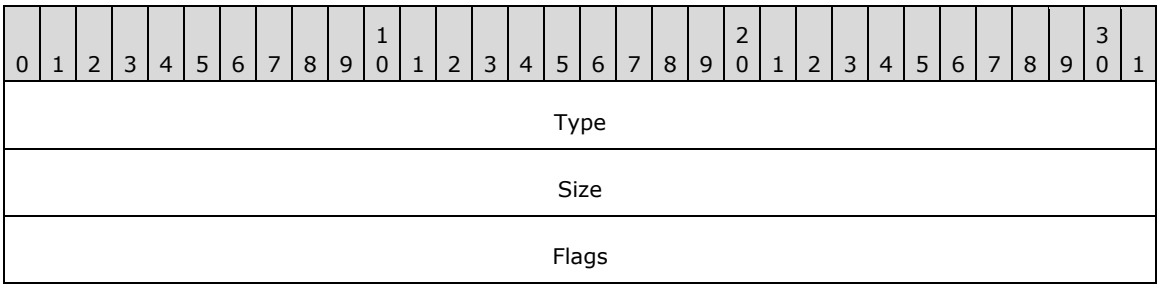
MM_ISOTROPIC mode ensures a 1:1 aspect ratio.

MM_ANISOTROPIC mode allows the x-coordinates and y-coordinates to be adjusted independently.

See section 2.3.11 for more state record types.

2.3.11.20 EMR_SETMAPPERFLAGS Record

The **EMR_SETMAPPERFLAGS** record specifies parameters for the process of matching logical fonts to physical fonts, which is performed by the font mapper.<88>



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETMAPPERFLAGS. This value is 0x00000010.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x0000000C.

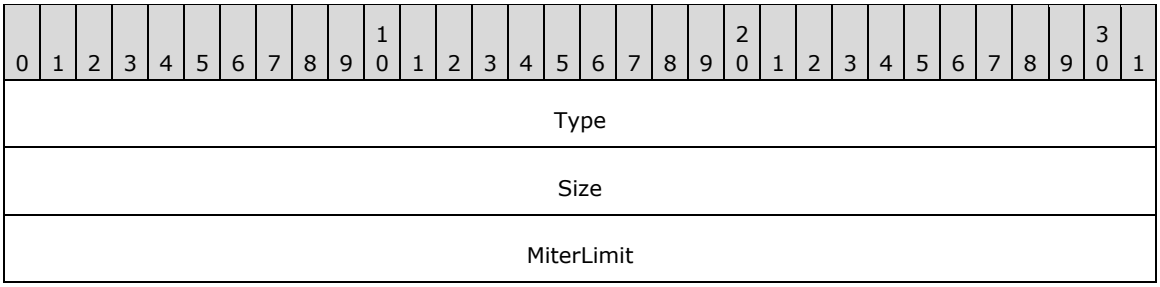
Flags (4 bytes): An unsigned integer that specifies parameters for the font matching process.

Value	Meaning
0x00000000	The font mapper is not limited to fonts that match the aspect ratio of the output device.
0x00000001	The font mapper SHOULD select only fonts that match the aspect ratio of the output device.

See section 2.3.11 for more state record types.

2.3.11.21 EMR_SETMITERLIMIT Record

The EMR_SETMITERLIMIT record specifies the limit for the length of miter joins.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETMITERLIMIT**. This value is 0x0000003A.

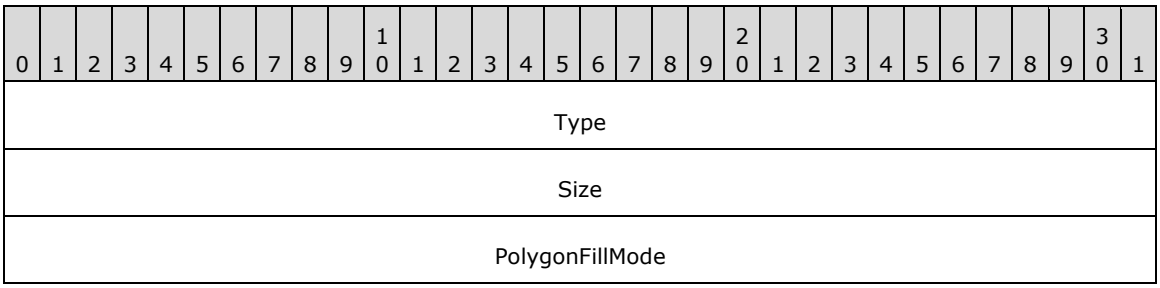
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

MiterLimit (4 bytes): An unsigned integer that specifies the new miter length limit.<89>

See section 2.3.11 for more state record types.

2.3.11.22 EMR_SETPOLYFILLMODE Record

The **EMR_SETPOLYFILLMODE** record defines polygon fill mode.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETPOLYFILLMODE. This value is 0x00000013.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

PolygonFillMode (4 bytes): An unsigned integer that specifies the polygon fill mode and is 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).

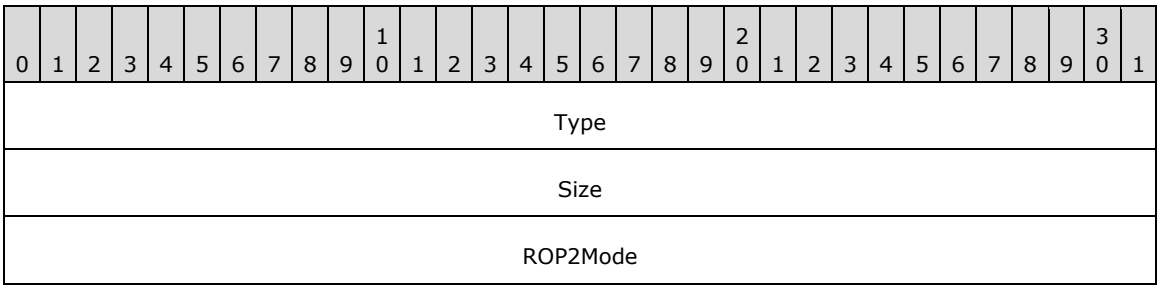
When the fill mode is ALTERNATE, the area between odd-numbered and even-numbered polygon sides on each scan line SHOULD be filled. That is, the area between the first and second side SHOULD be filled, and between the third and fourth side, and so on.

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 more state record types.

2.3.11.23 EMR_SETROP2 Record

The EMR_SETROP2 record defines a binary raster operation mode.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETROP2**. This value is 0x00000014.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

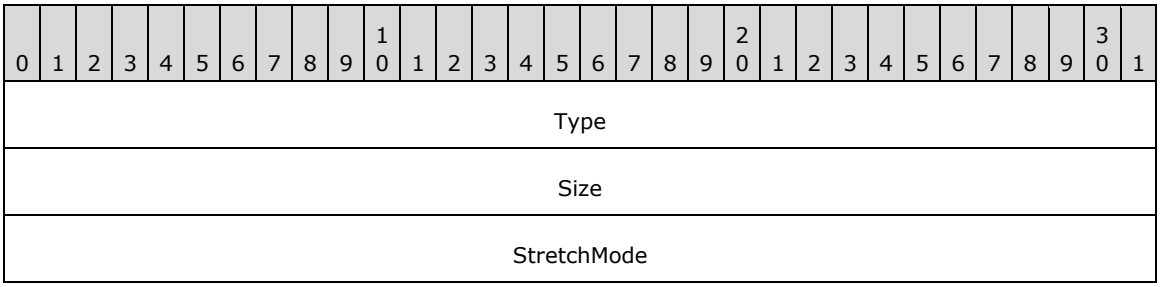
ROP2Mode (4 bytes): An unsigned integer that specifies the raster operation mode and is in the Binary Raster Op enumeration ([MS-WMF] section 2.1.1.2).

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 more state record types.

2.3.11.24 EMR_SETSTRETCHBLTMODE Record

The EMR_SETSTRETCHBLTMODE record specifies bitmap stretch mode.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETSTRETCHBLTMODE**. This value is 0x00000015.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

StretchMode (4 bytes): An 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.

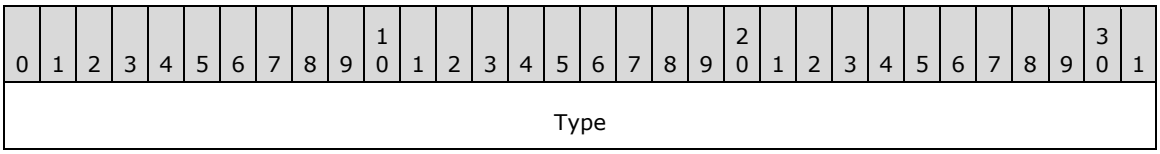
The **STRETCH_ANDSCANS** and **STRETCH_ORSCANS** modes are typically used to preserve foreground pixels in monochrome bitmaps. The **STRETCH_DELETESCANS** mode is typically used to preserve color in color bitmaps.

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 more state record types.

2.3.11.25 EMR_SETTEXTALIGN Record

The **EMR_SETTEXTALIGN** record specifies text alignment for text drawing.



Size
TextAlignmentMode

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETTEXTALIGN. This value is 0x00000016.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

TextAlignmentMode (4 bytes): An unsigned integer that specifies text alignment by using a mask of text alignment flags. These are either TextAlignmentMode flags ([MS-WMF] section 2.1.2.3) for text with a horizontal baseline, or 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_EXTTEXTOUTA, and EMR_EXTTEXTOUTW records (section 2.3.5) 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 drawing 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 more state record types.

2.3.11.26 EMR_SETTEXTCOLOR Record

The **EMR_SETTEXTCOLOR** record defines the current text foreground color.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Color																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETTEXTCOLOR. This value is 0x00000018.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Color (4 bytes): A ColorRef object ([MS-WMF] section 2.2.2.8) that specifies the text foreground color.

See section 2.3.11 for more 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.<90>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
nBreakExtra																															
nBreakCount																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETTEXTJUSTIFICATION. This value is 0x00000078.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

nBreakExtra (4 bytes): A signed integer that specifies the total amount of extra space to add in logical units.

nBreakCount (4 bytes): A signed integer that specifies the number of break characters.

Instead of using this record, an implementation SHOULD use EMR_EXTTEXTOUTW (section 2.3.5.8) to perform this function.

See section 2.3.11 for more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Extent																															
...																															

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETVIEWPORTEXTEX. This value is 0x0000000B.

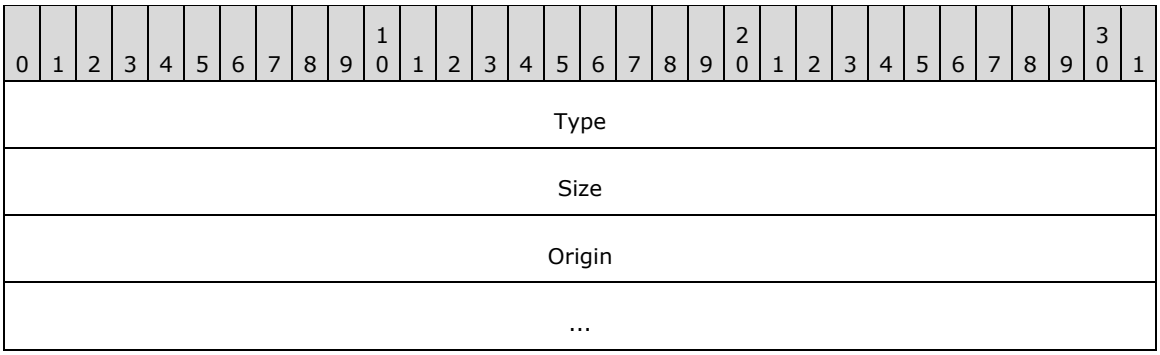
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Extent (8 bytes): A 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 more state record types.

2.3.11.29 EMR_SETVIEWPORTORGEX Record

The **EMR_SETVIEWPORTORGEX** record defines the viewport origin.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETVIEWPORTORGEX. This value is 0x0000000C.

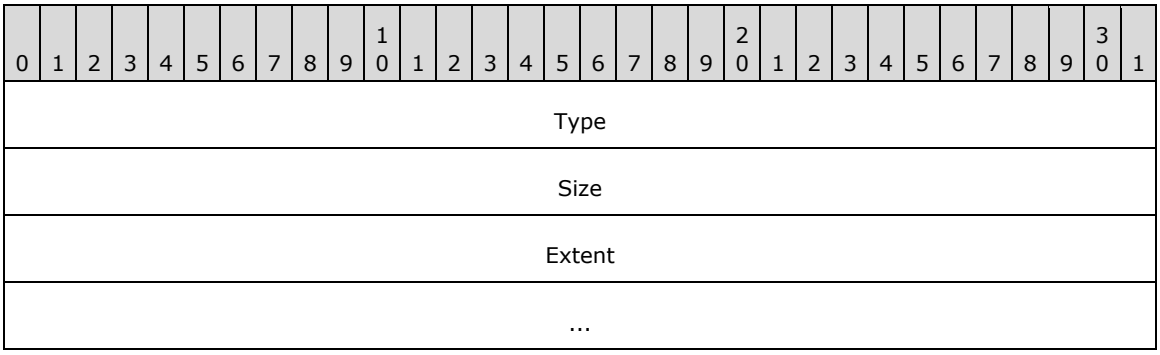
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Origin (8 bytes): A 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 more state record types.

2.3.11.30 EMR_SETWINDOWEXTEX Record

The **EMR_SETWINDOWEXTEX** record defines the window extent.



Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETWINDOWEXTEX. This value is 0x00000009.

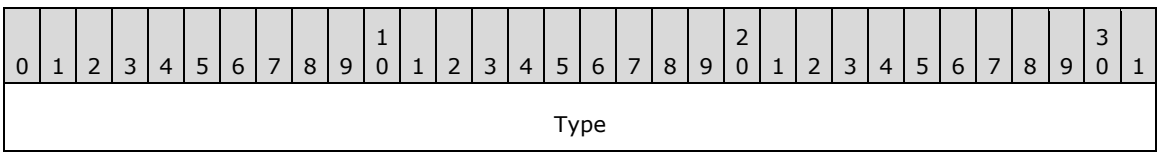
Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Extent (8 bytes): A 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 more state record types.

2.3.11.31 EMR_SETWINDOWORGEX Record

The **EMR_SETWINDOWORGEX** record defines the window origin.



Size
Origin
...

Type (4 bytes): An unsigned integer that identifies this record type as EMR_SETWINDOWORGEX. This value is 0x0000000A.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes.

Origin (8 bytes): A 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 more 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Xform (24 bytes)																															
...																															
...																															
...																															
TransformData (optional)																															

Type (4 bytes): An 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

Name	Value
EMR_MODIFYWORLDTRANSFORM	0x00000024

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

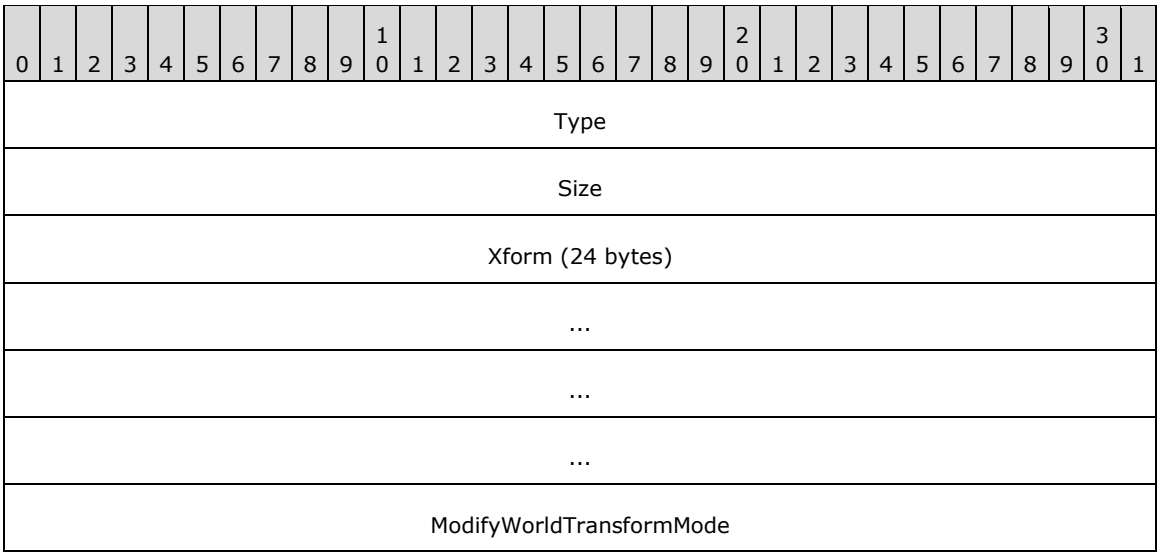
Xform (24 bytes): An XForm object (section 2.2.28), which defines a world-space to page-space transform.

TransformData (4 bytes, optional): An unsigned integer that specifies an additional parameter for the record.

See section 2.3 for more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_MODIFYWORLDTRANSFORM**. This value is 0x00000024.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 0x00000024.

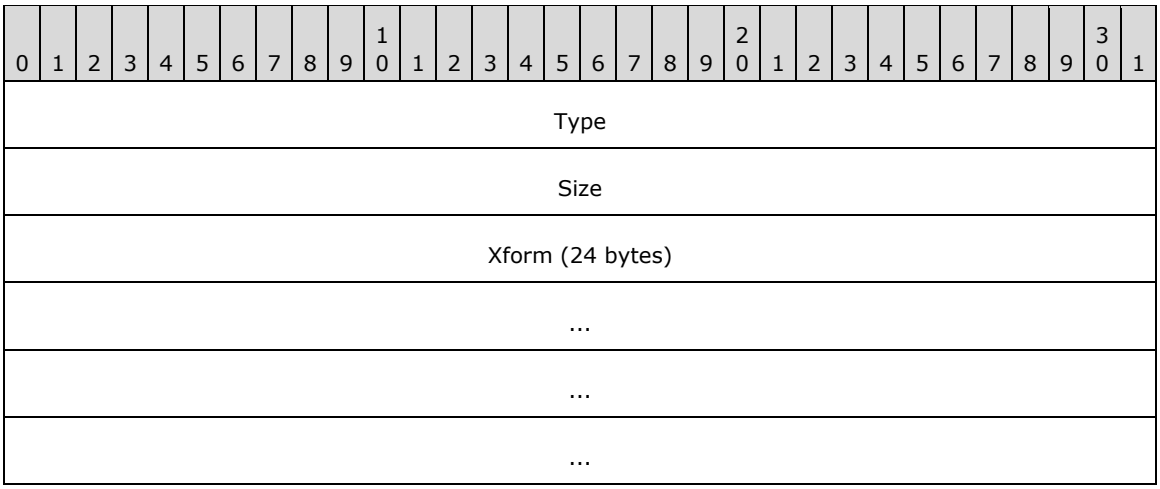
Xform (24 bytes): An XForm object (section 2.2.28) that defines a two-dimensional linear transform in logical units. This transform is used according to the **ModifyWorldTransformMode** to define a new value for the world-space to page-space transform in the playback device context.

ModifyWorldTransformMode (4 bytes): An unsigned integer that specifies how the transform specified in **Xform** is used. This value is 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 more 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.



Type (4 bytes): An unsigned integer that identifies this record type as **EMR_SETWORLDTRANSFORM**. This value is 0x00000023.

Size (4 bytes): An unsigned integer that specifies the size of this record in bytes. This value is 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 current world-space to page-space transform.

For more information concerning transforms and coordinate spaces, see [MSDN-WRLDPGSPC]. See section 2.3.12 for more transform record types.

3 Structure Examples

3.1 EMF Metafile Playback

This section describes the data and processing required for rendering the image stored in an EMF metafile. This process is referred to as "playback" throughout this specification. During metafile playback, the graphics state is maintained in the playback device context (section 3.1).

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains during the processing of this file format. The description of the organization is provided to facilitate the explanation of that processing. This document does not mandate that implementations adhere to this model, provided their external behavior is consistent with that specified in this document.

The following abstract data model elements are used to maintain the current state of the playback of an EMF metafile.

EMF object table (variable): A table of object metadata that is used to keep track of objects created and used during metafile playback. The **EMF object table** structure is described in section 3.1.1.1.

Graphics Environment (variable): The set of drawing parameters and metadata that specify the rendering of the image in the metafile. The **Graphics Environment** is described in section 3.1.1.2.

No part of the abstract data model is assumed to persist across system restarts; however, with this data it would be possible to recover the playback device context and resume metafile playback.

3.1.1.1 EMF Object Table

The **EMF object table** is an element of the state maintained during EMF metafile playback. It contains data used for managing graphics objects as they are created, activated, used, deactivated, and deleted by the processing of EMF records.

A possible implementation is described by the following elements.

Element name	Type	Set by
EMFObjectArray	**EMFObject[]**	Object creation record (section 2.3.7)
EMFObject	**EMFObject**	Object creation record

EMFObjectArray (variable): An array of **EMFObject** elements. The maximum number of objects is specified in the **Handles** field of the Header object (section 2.2.9) in an EMF_HEADER record (section 2.3.4.2). The table should be large enough for (**Handles** + 1) because element zero in the array is reserved.

EMFObject (variable): A graphics object and its associated index.

When a graphics object is created by an object creation record, 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 that 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 is instantiated and activated, a default stock object (section 2.1.31) for that type is used in graphics operations.

Element name	Type	Set by
GraphicsObjectIndex	UINT32	Object creation record
GraphicsObject	See element description.	Object creation record

GraphicsObjectIndex (4 bytes): An unsigned integer that specifies the index associated with the **GraphicsObject**, which is used to refer to the object after it is created.

GraphicsObject (variable): One of the following structures.

- DeviceIndependentBitmap ([MS-WMF] section 2.2.2.9)
- LogBrushEx (section 2.2.12)
- LogColorSpace ([MS-WMF] section 2.2.2.11)
- LogColorSpaceW ([MS-WMF] section 2.2.2.12)
- LogFont (section 2.2.13)
- LogFontExDv (section 2.2.15)
- LogPalette (section 2.2.17)
- LogPenEx (section 2.2.20)

The following process can be used to manage graphics objects with the **EMF object table**:

1. At the start of metafile processing, the **EMFObjectArray** structure is created.
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.

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.

3.1.1.2 Graphics Environment

The **Graphics Environment** describes the graphics state maintained during EMF metafile playback. A possible implementation is described by the following elements.

Element name	Type	Set by
PlaybackStateArray	**PlaybackState[]**	EMR_SAVEDC (section 2.3.11)
PlaybackState	UINT8[]	EMF records (section 2.3)

PlaybackStateArray: An array of saved **PlaybackState** elements, any of which can be used by the EMR_RESTOREDC record (section 2.3.11.6) to restore a previous graphics environment.

PlaybackState: The playback device context at ~~each~~some point in EMF record processing, including region definitions, color profiles, fonts and text properties, and graphics drawing metadata. The elements of the **PlaybackState** ~~structure contains elements that~~ are grouped as shown in the following table.

Element group	Section	Description
Regions	3.1.1.2.1	The current clipping regions and related properties
Colors	3.1.1.2.2	The current color profile and related properties
Text	3.1.1.2.3	Properties related to fonts and rendering text
Drawing	3.1.1.2.4	The graphics properties that determine how drawing commands render the image

The sections that follow describe the ~~types of the~~ elements ~~of~~in each group as well as the EMF records that set their values. Unreferenced intrinsic types are defined in [MS-DTYP].

3.1.1.2.1 Regions

The **Regions** group of elements control the output area and clipping properties in the playback device context.

Element name	Type	Set by
Clipping	RegionData (section 2.2.24)	EMR_EXCLUDECLIPRECT (section 2.3.2.1) EMR_EXTSELECTCLIPRGN (section 2.3.2.2) EMR_INTERSECTCLIPRECT (section 2.3.2.3) EMR_OFFSETCIPRGN (section 2.3.2.4)
MetaClipping	RegionData	EMR_SETMETARGN (section 2.3.2)
Viewport	UINT8[16]	EMR_SCALEVIEWPORTEXTEX (section 2.3.11.7) EMR_SETVIEWPORTEXTEX (section 2.3.11.28) EMR_SETVIEWPORTORGEX (section 2.3.11.29)
Window	UINT8[16]	EMR_SCALEWINDOWEXTEX (section 2.3.11.8) EMR_SETWINDOWEXTEX (section 2.3.11.30)

Element name	Type	Set by
		EMR_SETWINDOWORGEX (section 2.3.11.31)

Clipping: The current clipping region, which with **MetaClipping** defines the bounds of the drawing area. The default value for the **Clipping** element is implementation-specific.<91>

MetaClipping: The current metaregion, which with the **Clipping** region defines the bounds of the drawing area.

Viewport: A rectangular drawing area using coordinates in the device space.

Element name	Type	Set by
Extent	SizeL ([MS-WMF] section 2.2.2.22)	EMR_SCALEVIEWPORTEXTEX EMR_SETVIEWPORTEXTEX
Origin	PointL ([MS-WMF] section 2.2.2.15)	EMR_SETVIEWPORTORGEX

Extent: Horizontal and vertical sizes of the drawing area in device units.

Origin: Point value of the origin of the drawing area in device units.

Window: A rectangular drawing area using the coordinates of the page space.

Element name	Type	Set by
Extent	SizeL	EMR_SCALEWINDOWEXTEX EMR_SETWINDOWEXTEX
Origin	PointL	EMR_SETWINDOWORGEX

Extent: Horizontal and vertical sizes of the drawing area in logical units.

Origin: Point value of the origin of the drawing area in logical units.

3.1.1.2.2 Colors

The **Colors** group of elements define the current state of color management in the playback device context.

Element name	Type	Set by
ColorAdjustment	ColorAdjustment (section 2.2.2)	EMR_SETCOLORADJUSTMENT (section 2.3.11.13)
ColorProfile	UINT8[]	EMR_COLORMATCHTOTARGETW (section 2.3.11.1) EMR_SETICMPROFILEA (section 2.3.11.15) EMR_SETICMPROFILEW (section 2.3.11.16)
ColorProfileEmbedded	BOOL	EMR_COLORMATCHTOTARGETW
ColorProofing	UINT32	EMR_COLORMATCHTOTARGETW
ColorTransform	Implementation-dependent	EMR_COLORMATCHTOTARGETW
ICMMode	UINT32	EMR_SETICMMODE (section 2.3.11.14)
PixelFormat	PixelFormatDescriptor (section 2.2.22)	EMR_HEADER (section 2.3.4.2) EMR_PIXELFORMAT (section 2.3.11.5)

3.1.1.2.3 Text

The **Text** group of elements define the current font and text properties in the playback device context.

Text elements are used by the following EMF text drawing records.

- EMR_EXTTEXTOUTA (section 2.3.5.7)
- EMR_EXTTEXTOUTW (section 2.3.5.8)
- EMR_POLYTEXTOUTA (section 2.3.5.32)
- EMR_POLYTEXTOUTW (section 2.3.5.33)
- EMR_SMALLTEXTOUT (section 2.3.5.37)

Element name	Type	Set by
FontMapperFlags	UINT32	EMR_SETMAPPERFLAGS (section 2.3.11.20)
ForceUFIMapping	UniversalFontId (section 2.2.27)	EMR_FORCEUFIMAPPING (section 2.3.11.2)
LinkedUFIs	UniversalFontId[]	EMR_SETLINKEDUFIS (section 2.3.11.18)
TextAlignment	UINT32	EMR_SETTEXTALIGN (section 2.3.11.25)
TextJustification	UINT32 [2]	EMR_SETTEXTJUSTIFICATION (section 2.3.11.27)

3.1.1.2.4 Drawing

The **Drawing** group of elements define various graphics flags and other metadata values that affect how the image in the EMF metafile is rendered.

Element name	Type	Set by
ArcDirection	UINT32	EMR_SETARCDIRECTION (section 2.3.11.9)
BackgroundColor	ColorRef ([MS-WMF] section 2.2.2.8)	EMR_SETBKCOLOR (section 2.3.11.10)
BackgroundMode	UINT32	EMR_SETBKMODE (section 2.3.11.11)
BrushOrigin	PointL	EMR_SETBRUSHORGEX (section 2.3.11.12)
CurrentPosition	PointL ([MS-WMF] section 2.2.2.15)	EMR_MOVETOEX (section 2.3.11.4)
LayoutMode	UINT32	EMR_SETLAYOUT (section 2.3.11.17)
LineCap	UINT32	
LineJoin	UINT32	
MappingMode	UINT32	EMR_SETMAPMODE (section 2.3.11.19)
MiterLimit	UINT32	EMR_SETMITERLIMIT (section 2.3.11.21)
Path	PointL[]	Path bracket records (section 2.3.10)
PathBracket	Boolean	Path bracket records

Element name	Type	Set by
PolyFillMode	UINT32	EMR_SETPOLYFILLMODE (section 2.3.11.22)
ROP2	UINT32	EMR_SETROP2 (section 2.3.11.23)
StretchBLTMode	UINT32	EMR_SETSTRETCHBLTMODE (section 2.3.11.24)
TextColor	ColorRef	EMR_SETTEXTCOLOR (section 2.3.11.26)

ArcDirection: The drawing direction for arcs and rectangles, from the ArcDirection enumeration (section 2.1.2).

BackgroundColor: The color used as background for drawing text, hatched brushes, and pen styles that are not solid lines, depending on the **BackgroundMode** value.

BackgroundMode: How to combine the drawing background with the **BackgroundColor** value, from the BackgroundMode enumeration (section 2.1.4).

BrushOrigin: The horizontal and vertical origin of the current brush in logical units, which is used as needed to maintain an alignment of patterns on the display surface.

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, metafile playback is implemented so that it can read multi-byte values based on the endian method used by the computer that generated the output.

Because metafiles are developed and written on little-endian computers, playback performs this computation on computers that are big-endian.

3.2 EMF Metafile Example

This section describes an example of an EMF metafile, which when processed renders the following image.

Figure 4: 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 [ISO/IEC-8859-1]. The sections that follow describe the packets that convey this series of bytes.

```
00000000:01 00 00 00 D4 00 00 00 00 00 00 00 00 00 00 00 ....Ô...........
00000010:59 00 00 00 59 00 00 00 00 00 00 00 00 00 00 00 Y...Y...........
00000020:42 0C 00 00 41 0C 00 00 20 45 4D 46 00 00 01 00 B...A... EMF....
00000030:FC 37 00 00 16 00 00 00 05 00 00 00 34 00 00 00 ü7..........4...
00000040:6C 00 00 00 00 00 00 00 80 07 00 00 B0 04 00 00 l.......€...°...
00000050:A5 02 00 00 A7 01 00 00 00 00 00 00 00 00 00 00 ¥...§...........
00000060:00 00 00 00 D5 55 0A 00 A5 75 06 00 53 00 61 00 ....ÕU..¥u..S.a.
00000070:6D 00 70 00 6C 00 65 00 20 00 45 00 4D 00 46 00 m.p.l.e. .E.M.F.
00000080:20 00 74 00 68 00 61 00 74 00 20 00 68 00 61 00 .t.h.a.t. .h.a.
00000090:73 00 20 00 61 00 20 00 62 00 72 00 75 00 73 00 s. .a. .b.r.u.s.
000000A0:68 00 20 00 66 00 69 00 6C 00 6C 00 2C 00 20 00 h. .f.i.l.l.,. .
000000B0:62 00 69 00 74 00 6D 00 61 00 70 00 2C 00 20 00 b.i.t.m.a.p.,. .
000000C0:61 00 6E 00 64 00 20 00 74 00 65 00 78 00 74 00 a.n.d. .t.e.x.t.
000000D0:00 00 00 00 27 00 00 00 18 00 00 00 01 00 00 00 ....'...........
000000E0:02 00 00 00 52 47 2A 00 03 00 00 00 25 00 00 00 ....RG*.....%...
000000F0:0C 00 00 00 01 00 00 00 4C 00 00 00 64 00 00 00 ........L...D...
00000100:00 00 00 00 00 00 00 00 59 00 00 00 59 00 00 00 ........Y...Y...
00000110:00 00 00 00 00 00 00 00 5A 00 00 00 5A 00 00 00 ........Z...Z...
00000120:21 00 F0 00 00 00 00 00 00 00 00 00 00 00 80 3F !.ð...........€?
00000130:00 00 00 00 00 00 00 00 00 00 00 00 80 3F 00 00 ............€?..
00000140:00 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 25 00 00 00 ............%...
00000160:0C 00 00 00 00 00 00 00 80 4C 00 00 00 A8 2F 00 00 .......€L...¨/..
00000170:00 00 00 00 2D 00 00 00 59 00 00 00 59 00 00 00 ....-...Y...Y...
00000180:00 00 00 00 2D 00 00 00 5A 00 00 00 2D 00 00 00 ....-...Z...-...
00000190:20 00 CC 00 00 00 00 00 00 00 00 00 00 00 80 3F .Ì...........€?
000001A0:00 00 00 00 00 00 00 00 00 00 00 00 80 3F 00 00 ............€?..
000001B0:00 00 00 00 FF FF FF 00 00 00 00 00 64 00 00 00 ....ÿÿÿ.....D...
000001C0:28 00 00 00 8C 00 00 00 1C 2F 00 00 28 00 00 00 (...Œ..../..(...
000001D0:59 00 00 00 2D 00 00 00 01 00 18 00 00 00 00 00 Y...-...........
000001E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ./..............
000001F0:00 00 00 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E ....?_.?_.?_.?_.
00000200:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F ?_.?_.?_.>^.?_.?
00000210:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E _.A^.A^.A^.A^.A^
00000220:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A^.A^.A^.A^.A^.
00000230:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 3D A^.A^.A^.A^.@].=
00000240:59 0C 5A 60 4E AE AE AE BF BF BF C5 C5 C5 C0 C0 Y.Z`N®®®¿¿¿ÅÅÅÀÀ
00000250:C1 B1 B2 B1 A1 A1 A1 A9 AA AA AE AF B0 A2 A2 A2 Á±²±¡¡¡©ªª®¯°¢¢¢
```

```
00000260:A6 A6 A4 AF AE AD AC AC AC A6 A6 A6 99 99 99 7D ¦¦¤¯®-¬¬¬¦¦¦™™™}
00000270:7D 7D 66 66 65 5A 5A 59 4F 4F 4F 58 58 58 76 76 }}FFEZZYOOOXXXvv
00000280:76 9E 9E 9E B5 B5 B5 3C 43 2D 32 46 0A 34 4A 0A vžžžµµµ<C-2F.4J.
00000290:34 49 0A 36 4C 0B 3A 52 0B 3F 59 0C 41 5C 0D 42 4I.6L.:R.?Y.A\.B
000002A0:5D 0D 42 5D 0D 44 5D 0D 44 5D 0D 43 5C 0C 43 5C ].B].D].D].C\.C\
000002B0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C .C\.B[.C\.C\.C\.
000002C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
000002D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000002E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000002F0:45 5B 0C 45 5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00 E[.E[.E\.E\.E\..
00000300:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?_.?_.?_.?_.?_.?
00000310:5F 0E 3F 5F 0E 3E 0D 3F 5F 0E 3F 5F 0E 41 5E _.?_.>^.?_.?_.A^
00000320:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A^.A^.A^.A^.A^.
00000330:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 A^.A^.A^.A^.A^.A
00000340:5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 39 52 0C 70 73 ^.A^.A^.@].9R.ps.
00000350:6B AD AD AD A8 A8 A8 99 99 98 A6 A6 A6 A6 A7 A7 k---¨¨¨™™˜¦¦¦¦§§
00000360:A3 A5 A6 A6 A5 A3 AE A5 9D C9 BB AD A1 9B 95 A2 £¥¦¦¥£®¥□É»-¡›•¢
00000370:9F 9C B3 B1 AF B6 B4 B3 A9 A9 A9 97 93 90 8E 87 Ÿœ³±¯¶´³©©©--"□Ž‡
00000380:81 89 84 7F 81 7E 7A 79 78 76 80 7F 7E 98 98 98 □‰„□~zyxv€~˜˜˜
00000390:BC BC BC 6D 70 65 33 47 0A 37 4D 0B 35 4B 0A 35 ¼¼¼mpE3G.7M.5K.5
000003A0:4B 0B 38 4F 0B 3D 56 0C 41 5B 0D 42 5D 0D 42 5D K.8O.=V.A[.B].B]
000003B0:0D 43 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 43 5C 0C .C].D].C\.C\.C\.
000003C0:42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 B[.C\.C\.C\.C\.C
000003D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000003E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000003F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 C\.C\.C\.C\.C\.E[.E
00000400:5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F [.E\.E\.E\..?_.?
00000410:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F _.?_.?_.?_.?_.?_
00000420:0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E .>^.?_.?_.A^.A^.
00000430:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 A^.A^.A^.A^.A^.A
00000440:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E ^.A^.A^.A^.A^.A^
00000450:0E 41 5E 0E 40 5D 0D 38 51 0E 90 91 8E AC AC AC .A^.@].8Q.□'Ž¬¬¬
00000460:C3 C3 C3 C5 C6 C6 B9 BB BC B3 B0 AD BA AD 9E C8 ÃÃÃÅÆÆ¹»¼³°-°-žÈ
00000470:B4 A0 DB C8 B4 E4 D1 BE C2 B5 A9 82 7E 7A 8C 88 ´ ÛÈ´äÑ¾Âµ©‚~zŒˆ
00000480:84 83 80 7E 98 8F 85 BE AC 9A C9 B6 A3 CC BB A9 „ƒ€~˜□…¾¬šÉ¶£Ì»©
00000490:CD C0 B3 AB A3 9B 80 7E 76 78 76 70 91 8F 8D 84 ÍÀ³«£›€~vxvp'□□„
000004A0:86 80 36 4B 0B 39 50 0B 38 4E 0B 37 4D 0B 38 4F †€6K.9P.8N.7M.8O
000004B0:0B 3C 55 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D 0D .<U.@Z.B].B].B].
000004C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 C\.C\.C\.C\.B[.C
000004D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000004E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000004F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000500:5C 0C 43 5C 0C 43 5C 0C 45 5B 0A 45 5B 0A 45 5C \.C\.C\.E[.E[.E\
00000510:0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F .E\.E\..?_.?_.?_
00000520:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D .?_.?_.?_.?_.>^.
00000530:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 ?_.?_.A^.A^.A^.A
00000540:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E ^.A^.A^.A^.A^.A^
00000550:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A^.A^.A^.A^.A^.
00000560:40 5D 0D 3D 50 1C 94 94 94 A1 A1 A0 C9 CA CA BC @].=P.”””¡¡ ÉÊÊ¼
00000570:BB B8 BB AC 9C C4 AD 94 D2 BB A4 E0 CF BD E0 D0 »¸»¬œÄ-”Ò»¤àÏ½àÐ
00000580:BF DA C7 B5 DE CB B8 9A 93 8B 7C 76 70 A2 93 83 ¿ÚÇµÞË¸š"‹|vp¢"ƒ
00000590:C0 AB 95 C6 B2 9D CA B8 A4 D1 C0 AD E0 D2 C4 E8 À«•Æ²•Ê¸¤ÑÀ-àÒÄè
000005A0:DC CF CD C3 B5 8A 85 74 72 72 61 83 82 7E 34 43 ÜÏÍÃµŠ…trrAƒ‚~4C
000005B0:15 3C 54 0C 38 50 0B 38 4E 0B 39 50 0B 3D 55 0C .<T.8P.8N.9P.=U.
000005C0:40 5B 0D 42 5D 0D 42 5D 0D 42 5D 0D 43 5C 0C 43 @[.B].B].B].C\.C
000005D0:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C \.C\.C\.B[.C\.C\
000005E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000005F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000600:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000610:0C 43 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A .C\.E\.E\.E\.E\.
00000620:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E\..?_.?_.?_.?_.
00000630:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F ?_.?_.?_.>^.?_.?
00000640:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E _.A^.A^.A^.A^.A^
00000650:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A^.A^.A^.A^.A^.
00000660:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 4B A^.A^.A^.A^.@].K
00000670:56 33 B8 B8 B8 B7 B7 B7 A1 A1 A2 BC AC 9B CC B2 V3¸¸¸•••¡¡¢¼¬›Ì²
00000680:97 D6 C3 AE E5 D6 C6 E4 D4 C4 E1 D2 C2 D9 C6 B4 --ÖÃ®åÖÆäÔÄáÒÂÙÆ´
00000690:D7 C3 AD CE BF AF AD 9A 85 BB A5 8D BD A9 93 C3 ×Ã-Î¿¯-š…»¥•½©"Ã
000006A0:AF 9B C8 B5 A2 CE BD AA DC CE C0 E2 D5 C9 E1 D4 ¯›Èµ¢Î½ªÜÎÀâÕÉáÔ
000006B0:C7 D5 C5 B4 B1 A9 95 91 97 7B 4A 57 2D 39 51 0B ÇÕÅ´±©•'--{JW-9Q.
```

```
000006C0:38 50 0B 38 50 0B 3A 52 0C 3E 58 0C 41 5C 0D 42 8P.8P.:R.>X.A\.B
000006D0:5D 0D 42 5D 0D 42 5D 0D 41 5C 0C 43 5C 0C 43 5C ].B].B].A\.C\.C\
000006E0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C .C\.B[.C\.C\.C\.
000006F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
00000700:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\.C\
00000710:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A .C\.C\.C\.C\.C\.C].
00000720:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 00 E\.E\.E\.E\.E\.E\..
00000730:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?_.?_.?_.?_.?_.?_.?
00000740:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E _.?_.@].?_.?_.?_.A^
00000750:0F 41 5E 0F 41 5E 0F 41 5E 0E 41 5E 0E 40 5D 0D .A^.A^.A^.A^.A^.@].
00000760:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 @].@].@].A^.@].@].@
00000770:5D 0D 40 5D 0D 41 5E 0E 41 5E 0E 61 6C 44 D0 D0 ].@].A^.A^.AlDÐÐ
00000780:D0 B0 B0 B0 C8 C3 BE DB C8 B3 DC CA B8 EB DD CF Ð°°°ÈÃ¾ÛÈ³ÜÊ¸ëÝÏ
00000790:E8 D9 CA E5 D6 C7 E4 D6 C8 D9 C8 B5 D2 BF AA DD èÙÊåÖÇäÖÈÙÈµÒ¿ªÝ
000007A0:CC BA C7 B3 9E B6 9F 89 BC A7 92 C1 AD 98 C6 B3 Ì°Ç³ž¶Ÿ‰¼§'Á-˜Æ³
000007B0:9F CC BA A8 DB CD BF DD D0 C2 DF D3 C6 E0 D3 C5 ŸÌ°¨ÛÍ¿ÝÐÂßÓÆàÓÅ
000007C0:E2 D4 C4 A7 AA 8E 52 62 2B 37 4D 0B 39 51 0B 3B âÔÄ§ªŽRB+7M.9Q.;
000007D0:53 0C 3E 57 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D S.>W.@Z.B].B].B]
000007E0:0D 42 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 45 5B 0D .B].D].C\.C\.E[.
000007F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
00000800:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\.C\
00000810:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.C\.
00000820:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 C\.C\.C\.C\.E\.E\.E
00000830:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F \.E\.E\.E[..?_.?
00000840:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F _.?_.?_.?_.?_.?_.?_
00000850:0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0F .@].?_.?_.A^.A^.
00000860:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 A^.@].@].@].@].@
00000870:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].A^.@].@].@].@]
00000880:0D 41 5E 0E 3F 5B 0D 6A 79 47 D1 D2 CE C5 C2 BF .A^.?[.jyGÑÒÎÅÂ¿
00000890:EF E6 DD E7 D8 CA EE E0 D3 ED DF D2 EA DC CE E9 ïæÝçØÊîàÓíßÒêÜÎé
000008A0:DB CD E1 D1 C1 D6 C3 B0 D7 C4 B1 DA C9 B8 D5 C3 ÛÍáÑÁÖÃ°×Ä±ÚÉ¸ÕÃ
000008B0:B1 B8 A3 8C BC A7 91 C0 AC 97 C5 B1 9D CB B9 A5 ±¸£Œ¼§'À¬--Å±•Ë¹¥
000008C0:D6 C8 B8 CE BF AE D9 CB BD E1 D4 C7 E3 D4 C6 BC ÖÈ¸Î¿®ÙË½áÔÇãÔÆ¼
000008D0:B9 A2 53 63 2C 37 4C 0B 3C 55 0C 3E 58 0C 40 5B ¹¢Sc,7L.<U.>X.@[
000008E0:0D 41 5C 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D .A\.B].B].B].B].
000008F0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 C\.C\.C\.E[.C\.C
00000900:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\.C\
00000910:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.C\.
00000920:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
00000930:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C \.C\.E\.E\.E\.E\.E\
00000940:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F .E\.E[..?_.?_.?_.?_
00000950:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D .?_.?_.?_.?_.?_.@].
00000960:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 ?_.?_.A^.A^.@].@
00000970:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E ].@].@].@].@].A^
00000980:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.
00000990:40 59 0F 75 84 54 A6 B1 8E DC D9 CE F5 EE E8 E7 @Y.u„T¦±ŽÜÙÎõîèç
000009A0:DA CE E5 D7 C8 EB DD D0 EC DF D2 E4 D5 C4 D8 C8 ÚÎå×ÈëÝÐìßÒäÕÄØÈ
000009B0:B5 D9 CA B8 D9 CB BA D9 C8 B6 D9 C8 B6 C4 B0 9C µÙÊ¸ÙË°ÙÈ¶ÙÈ¶Ä°œ
000009C0:BA A5 90 BF AA 95 C3 AF 9B C9 B6 A3 C9 B9 A7 C7 °¥□¿ª•Ã¯›É¶£É¹§Ç
000009D0:B7 A6 D1 C2 B3 DC CF C0 E0 D2 C3 CE C4 B3 5B 69 •¦ÑÂ³ÜÏÀàÒÃÎÄ³[i
000009E0:33 3C 52 0D 40 5A 0D 41 5C 0D 42 5D 0D 42 5D 0D 3<R.@Z.A\.B].B].
000009F0:42 5D 0D 42 5D 0D 41 5C 0C 42 5D 0D 43 5C 0C 43 B].B].A\.B].C\.C
00000A00:5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C \.C\.E[.C\.C\.C\
00000A10:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.C\.
00000A20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
00000A30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\.C\
00000A40:0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A .E\.E\.E\.E\.E\.E\.
00000A50:45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E[..?_.?_.?_.?_.
00000A60:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F ?_.?_.?_.@].?_.?
00000A70:5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D _.A^.A^.@].@].@]
00000A80:0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D .@].@].@].A^.@].
00000A90:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 44 5C 12 8B @].@].@].A^.D\.‹
00000AA0:97 6A 93 A2 76 C8 CA B5 F5 EE E9 E8 DC CF E0 D0 --j"¢vÈÊµõîéèÜÏàÐ
00000AB0:C1 DF D2 C4 E1 BB A1 D4 AC 8E D5 B2 97 D3 A4 84 ÁßÒÄá»¡Ô¬ŽÕ²--Ó¤„
00000AC0:D3 A5 86 D7 C4 B0 D7 C5 B1 CF BB A6 C2 AE 99 C1 Ó¥†×Ä°×Å±Ï»¦Â®™Á
00000AD0:AD 98 C1 AE 99 C5 B0 9B C7 B5 A2 C8 B8 A7 D1 C2 -˜Á®™Å°›Çµ¢È¸§ÑÂ
00000AE0:B2 E0 D3 C6 E4 D8 CA E2 D5 C6 71 7B 4B 42 58 10 ²àÓÆäØÊâÕÆq{KBX.
00000AF0:42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 B].B].B].B].B].B
00000B00:5D 0D 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C ].A\.A\.C\.C\.C\
00000B10:0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .E[.C\.C\.C\.C\.
```

```
00000B20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000B30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000B40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A .C\.C\.C\.C\.E\.
00000B50:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 E\.E\.E\.E\.E[..
00000B60:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?_.?_.?_.?_.?_.?
00000B70:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E _.?_.@].?_.?_.A^
00000B80:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00000B90:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 @].@].A^.@].@].@
00000BA0:5D 0D 40 5D 0D 41 5E 0E 4B 60 17 A0 AB 83 92 A2 ].@].A^.K`. «ƒ'¢
00000BB0:73 AC B6 94 F4 ED E7 E2 CF BE DA CA BA D7 C4 B2 s¬¶"ôíçâÏ¾ÚÊº×Ä²
00000BC0:D0 86 58 CA 71 3D C9 69 32 C6 5D 24 C4 54 17 CC Ð†XÊq=Éi2Æ]$ÄT.Ì
00000BD0:86 5D D5 C5 B2 D2 BC A6 C9 B3 9D C2 AD 96 C1 AC †]ÕÅ²Ò¼¦É³•Â–-Á¬
00000BE0:97 C8 B1 9A D9 C8 B6 D3 C4 B4 D1 C3 B3 DE D1 C4 —È±šÙÈ¶ÓÄ´ÑÃ³ÞÑÄ
00000BF0:E0 D3 C5 E3 D5 C6 A3 A4 81 48 5E 13 42 5D 0D 42 àÓÅãÕÆ£¤•H^.B].B
00000C00:5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 41 5C ].B].B].B].B].A\
00000C10:0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .A\.C\.C\.C\.E[.
00000C20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000C30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000C40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00000C50:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 C\.C\.C\.E\.E\.E
00000C60:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F \.E\.E\.E[..?_.?
00000C70:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F _.?_.?_.?_.?_.?_
00000C80:0E 40 5D 0D 3E 5E 0D 3E 5E 0D 40 5D 0D 40 5D 0D .@].>^.>^.@].@].
00000C90:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00000CA0:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].A^.@].@].@].@]
00000CB0:0D 41 5E 0E 52 64 1F A4 AF 8A 95 A3 75 B3 AC 84 .A^.RD.¤¯Š•£u³¬„
00000CC0:DC AB 87 D7 A3 7D DB CD BD D2 A1 7F CD 76 42 CB Ü«‡×£}ÛÍ½Ò¡ÍvBË
00000CD0:71 3C C9 69 32 C7 62 2A C5 5B 20 C3 54 19 D8 AD q<Éi2ÇB*Å[ ÃT.Ø-
00000CE0:90 E2 D7 CA D3 BE A9 CA B6 A0 D0 BD AA D1 BD A9 •â×ÊÓ¾©Ê¶ Ð½ªÑ½©
00000CF0:DD CD BD DD D0 C2 D0 C1 B1 DA CD BE DE D1 C3 DB ÝÍ½ÝÐÂÐÁ±ÚÍ¾ÞÑÃÛ
00000D00:CD BC 91 9A 6E 4E 62 18 42 5D 0D 42 5D 0D 42 5D Í¼'šnNB.B].B].B]
00000D10:0D 42 5D 0D 42 5E 0B 41 5C 0C 41 5C 0C 41 5C 0C .B].B^.A\.A\.A\.
00000D20:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 C\.C\.C\.E[.C\.C
00000D30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000D40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00000D50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000D60:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C \.C\.E\.E\.E\.E\
00000D70:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F .E\.E[..?_.?_.?_
00000D80:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D .?_.?_.?_.?_.@].
00000D90:3E 5D 0D 3E 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 >].>].@].@].@].@
00000DA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00000DB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0E .@].@].@].@].@].
00000DC0:58 67 26 A4 B1 8B B8 B1 8A D6 9F 74 D5 98 6B D7 Xg&¤±‹¸±ŠÖŸtÕ˜k×
00000DD0:A4 7F DD CF BF D2 90 66 CD 77 43 CA 6E 38 C8 65 ¤ÝÏ¿Ò•FÍwCÊn8ÈE
00000DE0:2C C6 60 27 C6 5B 21 C3 56 1B CB 71 40 DD C8 B6 ,Æ`'Æ[!ÃV.Ëq@ÝÈ¶
00000DF0:D5 C3 AF E1 D2 C2 DF D0 C1 D9 C8 B7 E1 D3 C4 E4 ÕÃ¯áÒÂßÐÁÙÈ•áÓÄä
00000E00:D8 CA D1 C2 B2 E0 D4 C7 E7 DA CD AD AA 90 6F 7F ØÊÑÂ²àÔÇçÚÍ-ª•o
00000E10:47 52 63 1F 41 5B 0D 41 5C 0C 41 5C 0C 42 5D 0D GRC.A[.A\.A\.B].
00000E20:42 5D 0B 41 5C 0C 41 5C 0C 41 5B 0C 42 5B 0C 43 B].A\.A\.A[.B[.C
00000E30:5B 0C 43 5C 0C 43 5B 0B 43 5C 0C 43 5C 0C 43 5B [.C\.C[.C\.C\.C[
00000E40:0C 43 5B 0C 43 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C .C[.C[.C\.C\.C\.
00000E50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00000E60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000E70:0C 44 5C 0A 44 5C 0A 45 5B 0A 45 5B 0A 45 5B 0A .D\.D\.E[.E[.E[.
00000E80:45 5B 0B 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E[..?_.?_.?_.?_.
00000E90:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 ?_.?_.?_.@].@].@
00000EA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00000EB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.
00000EC0:40 5D 0D 40 5D 0D 40 5D 0D 42 5C 0E 60 6E 33 A9 @].@].@].B\.`n3©
00000ED0:B6 91 CD B0 8B D7 A2 78 D6 99 6D D9 A8 86 E2 D7 ¶'Í°‹×¢xÖ™mÙ¨†â×
00000EE0:CA DE BC A2 CC 78 44 D0 85 57 D2 8B 61 CC 74 42 ÊÞ¼¢ÌxDÐ…WÒ‹AÌtB
00000EF0:CC 76 46 CD 7D 4F CC 76 47 CF 9C 7A D8 C9 B7 DF ÌvFÍ}OÌvGÏœzØÉ•ß
00000F00:D0 C0 DF CF BF DD CE BE E4 D7 C9 EB E0 D4 D1 C3 ÐÀßÏ¿ÝÎ¾ä×ÉëàÔÑÃ
00000F10:B3 DC CF C1 F1 E6 DA C8 C0 AE 72 80 4F 56 66 24 ³ÜÏÁñæÚÈÀ®r€OVF$
00000F20:42 5B 0E 41 5C 0C 41 5C 0C 42 5D 0D 43 5C 0C 43 B[.A\.A\.B].C\.C
00000F30:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\.C[.BZ.C[.C\
00000F40:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D .B[.C\.C\.C[.C[.
00000F50:43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C[.C\.C\.C\.C\.C
00000F60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00000F70:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
```

```
00000F80:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 C\.E[.E[.E[.E\..
00000F90:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?_.?_.?_.?_.?_.?
00000FA0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D _.?_.@].@].@].@]
00000FB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00000FC0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 @].@].@].A^.@].@
00000FD0:5D 0D 40 5D 0D 43 5C 10 78 82 4C B9 BF 9C D8 AB ].@].C\.x‚L¹¿œØ«
00000FE0:84 D6 A3 7A D6 A2 7C DE BE A5 E7 DA CD E8 DA CB „Ö£zÖ¢|Þ¾¥çÚÍèÚË
00000FF0:DA AA 89 E7 D9 CC DB AE 91 C7 63 2B C6 5D 24 C4 Úª‰çÙÌÛ®'ÇC+Æ]$Ä
00001000:56 1B C2 51 15 D1 A2 83 E1 D4 C5 E1 D2 C2 E1 D1 V.ÂQ.Ñ¢ƒáÔÅáÒÂáÑ
00001010:C2 D1 C0 AD D1 C2 B1 E2 D6 C9 D3 C5 B5 D4 C6 B7 ÂÑÀ-ÑÂ±âÖÉÓÅµÔÆ·
00001020:E2 D6 C8 EA DE D2 95 9A 76 5B 6B 2A 44 5C 0F 41 âÖÈêÞÒ•šv[k*D\.A
00001030:5C 0C 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C \.A\.A\.C\.C\.C\
00001040:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B .C[.BZ.C[.C\.B[.
00001050:43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43 C\.C\.C[.C[.C[.C
00001060:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001070:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00001080:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 C\.C\.C\.C\.C\.E
00001090:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F [.E[.E[.E\..?_.?
000010A0:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F _.?_.?_.?_.?_.?_
000010B0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000010C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000010D0:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D ].@].A^.@].@].@]
000010E0:0D 47 5E 13 8E 97 67 C4 BF 9F CE A9 81 CA B7 99 .G^.Ž--gÄ¿ŸÎ©□Ê•™
000010F0:E4 DA CF DB CC BC E9 DA CC E8 DA CC E9 DB CE E5 äÚÏÛÌ¼éÚÌèÚÌéÛÎå
00001100:CF BE CB 73 40 C6 5F 26 C6 5C 22 C3 52 15 CA 75 Ï¾Ës@Æ_&Æ\"ÃR.Êu
00001110:45 D9 CA B9 E6 D8 CA E3 D5 C6 D1 BF AE C3 AF 9A EÙÊ¹æØÊãÕÆÑ¿®Ã¯š
00001120:CE BE AD D1 C3 B3 CC BD AD D2 C4 B5 D9 CC BD DD Î¾-ÑÃ³Ì½-ÒÄµÙÌ½Ý
00001130:CE BF CE C6 B3 6A 78 3A 47 5C 11 41 5C 0C 41 5C Î¿ÎÆ³jx:G\.A\.A\
00001140:0C 42 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D .B\.C\.C\.C\.C[.
00001150:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 BZ.C[.C\.B[.C\.C
00001160:5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43 5C 0C 43 5C \.C[.C[.C[.C\.C\
00001170:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00001180:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
00001190:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B \.C\.C\.C\.C\.E[.E[
000011A0:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F .E[.E\..?_.?_.?_
000011B0:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D .?_.?_.?_.?_.@].
000011C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000011D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000011E0:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 4C 61 17 .A^.@].@].@].LA.
000011F0:9C A5 7A D1 BF 9E D6 A5 7D E2 BD 9F E3 CF BE DB œ¥zÑ¿žÖ¥}â½ŸãÏ¾Û
00001200:CB BB E9 DA CC E8 D9 CB EA DF D3 DC B0 93 C7 63 Ë»éÚÌèÙËêßÓÜ°"ÇC
00001210:29 C5 5C 22 C5 59 1F C5 5B 21 DD BC A4 DE D2 C4 )Å\"ÅY.Å[!Ý¼¤ÞÒÄ
00001220:E6 D8 CA E8 DA CC CD BB A9 C6 B3 9F CE BD AB CA æØÊèÚÌÍ»©Æ³ŸÎ½«Ê
00001230:B8 A5 CF BE AC D5 C4 B2 BD B4 9D AB AC 8B B6 BA ¸¥Ï¾¬ÕÄ²½´□«¬‹¶°
00001240:9A 78 86 4D 4A 5E 14 43 5C 0C 43 5C 0C 43 5C 0C šx†MJ^.C\.C\.C\.
00001250:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 C\.C\.C\.C[.BZ.C
00001260:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B [.C\.B[.C\.C\.C[
00001270:0D 42 5A 0C 43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C .BZ.C[.C\.C\.C\.
00001280:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00001290:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000012A0:0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C .C\.C\.E[.E[.E[.
000012B0:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E\..?_.?_.?_.?_.
000012C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 ?_.?_.?_.@].@].@
000012D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000012E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.
000012F0:40 5D 0D 40 5D 0D 40 5D 0D 53 64 1D A2 AA 82 D7 @].@].@].Sd.¢ª‚×
00001300:BC 99 D8 A7 7F DA A8 81 E3 CE BB E1 D2 C3 E8 D9 ¼™Ø§Ú¨•ãÎ»áÒÃèÙ
00001310:CC E8 D9 CB E4 D6 C7 E6 D7 C9 E2 BC A3 D2 84 57 ÌèÙËäÖÇæ×Éâ¼£Ò„W
00001320:C5 56 1A D4 9A 76 E6 DE D2 E0 D0 C1 E3 D5 C6 F0 ÅV.Ôšvæ ÞÒàÐÁãÕÆð
00001330:E4 D7 E1 D2 C3 CA B7 A4 D1 C1 B0 C9 B7 A3 D0 BD ä×áÒÃÊ·¤ÑÁ°É·£Ð½
00001340:A9 D8 C6 B2 B5 AC 92 7B 89 53 75 89 4D 70 82 46 ©ØÆ²µ¬'{‰Su‰Mp‚F
00001350:53 61 1C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 SA.C\.C\.C\.C\.C
00001360:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\.C[.BZ.C[.C\
00001370:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C .B[.C\.C\.C[.BZ.
00001380:42 5A 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 BZ.C\.C\.C\.C\.C
00001390:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000013A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000013B0:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 C\.E[.E[.E[.E\..
000013C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?_.?_.?_.?_.?_.?
000013D0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D _.?_.@].@].@].@]
```

```
000013E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000013F0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 @].@].@].A^.@].@
00001400:5D 0D 41 5D 0E 59 67 24 A9 B2 8C D1 BB 98 D7 A5 ].A].Yg$©²ŒÑ»˜×¥
00001410:7C D4 B0 8D DE D4 C5 DA CA BB DE D1 C5 DB D0 C3 |Ô°•ÞÔÅÚÊ»ÞÑÅÛÐÃ
00001420:DC CE BF E4 D5 C5 EC E2 D7 F0 E5 D9 E4 C4 AE D7 ÜÎ¿äÕÅìâ×ðåÙäÄ®×
00001430:C8 B9 D4 C7 BA D8 C9 BB E0 D2 C4 E9 DD D0 E3 D4 È¹ÔÇºØÉ»àÒÄéÝÐãÔ
00001440:C6 CF BD AB D2 C3 B3 C2 B2 A1 C6 B6 A3 CD BA A7 ÆÏ½«ÒÃ³Â²¡Æ¶£Íº§
00001450:CD BD A7 8B 97 65 7D 8E 55 74 85 49 5A 66 21 44 Í½§‹--E}ŽUt…IZF!D
00001460:5C 0E 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001470:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B .C[.BZ.C[.C\.B[.
00001480:43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 42 5A 0C 42 C\.C\.C[.BZ.BZ.B
00001490:5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C [.C\.C\.C\.C\.C\
000014A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000014B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 C\.C\.C\.C\.C\.E
000014C0:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F [.E[.E[.E\..?_.?
000014D0:5F 0E 3F 5F 0C 3F 5F 0C 3F 5F 0C 3F 5F 0E 40 5E _.?_.?_.?_.?_.@^
000014E0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000014F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001500:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 43 5D ].@].@].@].@].C]
00001510:0F 69 71 33 B3 BC 99 CD BA 96 D6 A4 79 A2 9B 72 .iq3³¼™Íº–Ö¤y¢›r
00001520:B3 AE 91 DE D0 C2 D9 C2 AF CE 9D 7C CF 8E 64 CE ³®'ÞÐÂÙÂ¯Î□|ÏŽDÎ
00001530:7D 4C D6 AB 8D DD D4 C9 DB D1 C6 DB CF C2 D8 CA }LÖ«•ÝÔÉÛÑÆÛÏÂØÊ
00001540:BC CF C1 B2 D1 C4 B6 DB CF C3 D6 C8 BB C6 B6 A5 ¼ÏÁ²ÑÄ¶ÛÏÃÖÈ»Æ¶¥
00001550:CF C2 B4 D4 C8 BC A1 9B 81 69 6C 3C 60 63 2E 6C ÏÂ´ÔÈ¼¡›•il<`C.l
00001560:77 41 7A 88 54 74 83 4B 60 6C 27 45 5C 0E 43 5C wAz^TtƒK`l'E\.C\
00001570:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D .C\.C\.C\.C\.C[.
00001580:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 BZ.C[.C\.B[.C\.C
00001590:5C 0C 43 5B 0D 44 5A 0C 44 5A 0C 44 5B 0B 43 5C \.C[.DZ.DZ.D[.C\
000015A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000015B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
000015C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0C 43 5B \.C\.C\.C\.C[.C[
000015D0:0C 45 5B 0C 43 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 .E[.C\..?_.?_.?`
000015E0:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D .?`.?`.?_.A^.@].
000015F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001600:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00001610:0D 40 5D 0D 40 5D 0D 40 5D 0D 45 5E 10 7D 82 48 .@].@].@].E^.},H
00001620:BE C5 A4 B9 B0 8A BA B2 93 D1 C9 B6 DB AC 89 E6 ¾Å¤¹°Šº²"ÑÉ¶Û¬‰æ
00001630:CC B7 E8 D2 C0 D1 85 55 CC 75 41 CA 6C 36 C6 6E Ì•èÒÀÑ…UÌuAÊl6Æn
00001640:3B B5 A4 8E D2 C6 B8 E7 D8 C9 E5 D6 C7 E0 D1 C2 ;µ¤ŽÒÆ¸çØÉåÖÇàÑÂ
00001650:DA CC BD DB CD C0 D3 C6 B7 BE AF A0 D0 C3 B5 E9 ÚÌ½ÛÍÀÓÆ•¾¯ ÐÃµé
00001660:DE D2 EA E2 D6 C3 BE A9 8B 89 64 61 66 34 65 6D ÞÒêâÖÃ¾©‹‰DAF4Em
00001670:3E 68 76 45 63 6C 2D 47 5D 0F 43 5C 0C 43 5C 0C >hvEcl-G].C\.C\.
00001680:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 C\.C\.C\.C[.BZ.C
00001690:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B [.C\.B[.C\.C\.C[
000016A0:0D 45 5B 0D 45 5B 0C 45 5B 0C 43 5C 0C 42 5B 0B .E[.E[.E[.C\.B[.
000016B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
000016C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000016D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .C\.C\.C\.C\.E[.
000016E0:43 5D 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C C]..?_.?_.?`.?`.
000016F0:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 ?`.?_.A^.@].@].@
00001700:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00001710:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00001720:40 5D 0D 40 5D 0D 47 5E 12 91 93 61 BC C2 A1 C3 @].@].G^.'"A¼Â¡Ã
00001730:C4 AB E8 E0 D4 DF B4 93 D4 96 68 DE B2 93 E0 BB Ä«èàÔß´"Ô–hÞ²"à»
00001740:A0 CF 81 50 CD 7A 48 CB 71 3C C8 63 2A D3 9A 75 Ï□PÍzHËq<È*Ósu
00001750:E1 D5 C7 DB CB BB D8 C7 B7 D7 C6 B5 D8 C7 B6 D8 áÕÇÛË»ØÇ•×ÆµØÇ¶Ø
00001760:C7 B6 D3 C1 AF C5 B2 9E DB CD BE E9 DE D3 EF E5 Ç¶ÓÁ¯Å²žÛÍ¾éÞÓïå
00001770:DB F6 EE E4 FA F0 E7 E3 D8 CA BB B0 9A A1 9D 83 Ûöîäúðçã؊»°š¡□ƒ
00001780:71 74 3B 4A 5E 12 43 5C 0C 43 5C 0C 43 5C 0C 43 qt;J^.C\.C\.C\.C
00001790:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\.C[.BZ.C[.C\
000017A0:0C 42 5B 0B 43 5B 0C 43 5B 0C 43 5C 0C 45 5B 0C .B[.C[.C[.C\.E[.
000017B0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 DZ.DZ.B[.B[.C\.C
000017C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
000017D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000017E0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0A 00 C\.C\.C\.E[.C\..
000017F0:3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F ?_.?_.?`.?`.?`.?
00001800:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D _.A^.@].@].@].@]
00001810:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00001820:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001830:5D 0D 4E 60 17 9B 9C 6F BB C2 9E D7 D5 C2 E0 C0 ].N`.›œo»Âž×ÕÂàÀ
```

```
00001840:A6 D7 A0 76 D6 9C 72 D8 A1 7A D4 95 6A D0 82 51 ¦× vÖœrØ¡zÔ•jÐ‚Q
00001850:CD 7A 47 CA 6D 38 D7 A7 88 DB BF AA D9 C9 B8 D3 ÍzGÊm8×§ˆÛ¿ªÙÉ¸Ó
00001860:C3 B2 D0 BF AC D0 BE AC D1 C0 AE CF BD AA C6 B3 Ã²Ð¿¬Ð¾¬ÑÀ®Ï½ªÆ³
00001870:9F D1 BF AD E2 D5 C7 E6 DC D0 EA DF D4 E9 DE D3 ŸÑ¿-âÕÇæÜÐêßÔéÞÓ
00001880:E4 D8 CC DC CE C0 D1 C0 AF CC BD AA 73 79 3E 50 äØÌÜÎÀÑÀ¯Ì½ªsy>P
00001890:5F 17 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C _.C\.C\.C\.C\.C\
000018A0:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B .C[.BZ.C[.C\.B[.
000018B0:43 5B 0D 43 5B 0D 43 5C 0C 45 5B 0C 44 5A 0B 44 C[.C[.C\.E[.DZ.D
000018C0:5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C Z.B[.B[.C\.C\.C\
000018D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000018E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C\.C
000018F0:5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F \.C\.E[.E\..?_.?
00001900:5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E _.?`.?`.?`.?_.A^
00001910:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00001920:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001930:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 57 64 ].@].@].@].@].Wd
00001940:1E A5 A5 7C B8 BE 9B D4 D1 BF E2 C5 AD D8 A2 78 .¥¥|¸¾›ÔÑ¿âÅ-Ø¢x
00001950:D7 9E 74 D8 A4 7E D2 8C 5E D0 83 53 CC 74 40 D1 ×žtØ¤~ÒŒ^Ðƒ SÌt@Ñ
00001960:8D 62 DE D0 C2 DC CE C0 D4 C2 B1 CF BD AB D1 BF •bÞÐÂÜÎÀÔÂ±Ï½«Ñ¿
00001970:AD D4 C3 B2 D4 C3 B1 CF BE AB D1 BF AD DC CC BC -ÔÃ²ÔÃ±Ï¾«Ñ¿-ÜÌ¼
00001980:DE D0 C3 DF D3 C6 DD D0 C3 DC CF C1 DF D3 C5 DA ÞÐÃßÓÆÝÐÃÜÏÁßÓÅÚ
00001990:CC BF D3 C5 B6 90 8D 58 60 6D 27 56 64 1E 43 5C Ì¿ÓÅ¶••X`m'Vd.C\
000019A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D .C\.C\.C\.C\.C[.
000019B0:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 43 BZ.C[.C\.B[.C[.C
000019C0:5B 0D 43 5C 0C 44 5A 0B 44 5A 0B 44 5A 0B 42 5B [.C\.DZ.DZ.DZ.B[
000019D0:0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .B[.C\.C\.C\.C\.
000019E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
000019F0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001A00:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 .E[.E\..?_.?_.?`
00001A10:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D .?`.?`.?_.A^.@].
00001A20:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001A30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00001A40:0D 40 5D 0D 40 5D 0D 40 5D 0D 62 6B 26 B0 B1 8C .@].@].@].Bk&°±Œ
00001A50:BB BE A0 E3 DD D0 E8 D7 C8 D8 A2 7A D7 A4 7D D5 »¾ ãÝÐè×ÈØ¢z×¤}Õ
00001A60:9C 74 D1 87 57 CE 7C 4A D0 88 5C D7 C2 AF D4 C5 œtÑ‡WÎ|JÐˆ\×Â¯ÔÅ
00001A70:B5 D5 C4 B2 D9 C9 B8 DD CD BD DC CC BC DA C9 B9 µÕÄ²ÙÉ¸ÝÍ½ÜÌ¼ÚÉ¹
00001A80:D7 C6 B6 D7 C6 B5 DF D0 C0 D7 C7 B6 D7 C8 B9 D5 ×Æ¶×ÆµßÐÀ×Ç¶×È¹Õ
00001A90:C7 B9 D9 CB BD E5 DA CD E4 D9 CC E0 D3 C6 D7 CD Ç¹ÙË½åÚÍäÙÌàÓÆ×Í
00001AA0:BD 7C 84 48 5B 6A 1F 5C 67 21 43 5C 0C 43 5C 0C ½|„H[j.\g!C\.C\.
00001AB0:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 C\.C\.C\.C\.C[.BZ.C
00001AC0:5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 42 5A 0C 42 5B [.C\.B[.C[.BZ.B[
00001AD0:0B 44 5A 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B .DZ.DZ.DZ.B[.B[.
00001AE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00001AF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001B00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .C\.C\.C\.C\.E[.
00001B10:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C E\..?_.?_.?`.?`.
00001B20:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 ?`.?_.A^.@].@].@
00001B30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00001B40:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00001B50:40 5D 0D 41 5D 0D 6D 73 32 B9 BA 9B D6 D1 BF EC @].A].ms2¹º›ÖÑ¿ì
00001B60:E3 DA EE E7 DF E1 C0 A5 D7 AB 88 D7 A3 7E DA A7 ãÚîçßáÀ¥×«ˆ×£~Ú§
00001B70:84 DD B1 94 D9 C4 B1 D4 C5 B4 DB CB BB E9 DB CD „Ý±"ÙÄ±ÔÅ´ÛË»éÛÍ
00001B80:E9 DB CE E3 D5 C6 DE CF BF DC CD BC DF CF C0 E1 éÛÎãÕÆÞÏ¿ÜÍ¼ßÏÀá
00001B90:D1 C2 DF CF C0 DA CA B9 D6 C7 B8 D5 C7 B9 E8 DC ÑÂßÏÀÚÊ¹ÖÇ¸ÕÇ¹èÜ
00001BA0:D1 E8 DC D1 E4 D8 CC E5 D9 CC B2 AD 93 7D 89 53 ÑèÜÑäØÌåÙÌ²-"}‰S.
00001BB0:6A 77 32 60 6A 24 44 5C 0C 43 5C 0C 43 5C 0C 43 jw2`j$D\.C\.C\.C
00001BC0:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\.C[.BZ.C[.C\
00001BD0:0C 42 5B 0B 42 5A 0C 42 5A 0C 42 5B 0B 44 5A 0B .B[.BZ.BZ.B[.DZ.
00001BE0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 DZ.DZ.B[.B[.C\.C
00001BF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001C00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00001C10:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 C\.C\.C\.E[.E\..
00001C20:40 5D 0D 41 5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F @].A^.?_.?_.?_.?
00001C30:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D _.A^.@].@].@].@]
00001C40:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .A^.@].@].@].@].
00001C50:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 A^.@].@].@].@].A
00001C60:5E 0F 77 79 3C CC CA B4 EB E2 D9 ED E4 DB EF E7 ^.wy<ÌÊ´ëâÙíäÛïç
00001C70:E0 EF E7 DE DE CC BB DD CD BC E8 DB CD E9 DC CF àïçÞÞÌ»ÝÍ¼èÛÍéÜÏ
00001C80:DC CD BD E5 D6 C7 F1 E4 D8 F0 E3 D6 EA DC CE E5 ÜÍ½åÖÇñäØðãÖêÜÎå
00001C90:D6 C7 E1 D2 C2 DF CF C0 DF CF C0 DD CD BD D9 C8 ÖÇáÒÂßÏÀßÏÀÝÍ½ÙÈ
```

```
00001CA0:B8 D7 C6 B5 E0 D3 C6 E6 DB CF EB E0 D5 E1 D5 C8 ¸×ÆµàÓÆæÛÏëàÕáÕÈ
00001CB0:DF D3 C7 E4 D8 CB A7 A3 88 77 82 4F 77 86 4B 6B ßÓÇäØË§£ˆw‚Ow†Kk
00001CC0:71 30 44 5D 0E 41 5C 0C 43 5C 0C 43 5C 0C 43 5C q0D].A\.C\.C\.C\
00001CD0:0C 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D .C\.C\.C[.C[.C[.
00001CE0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 B[.B[.B[.B[.DZ.D
00001CF0:5A 0B 42 5B 0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C Z.B[.B[.DZ.C\.C\
00001D00:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C .C].C].C].C\.C\.
00001D10:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\.C\.C
00001D20:5C 0C 43 5C 0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 \.C\.B[.D[..@].A
00001D30:5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E ^.?_.?_.?_.?_.A^
00001D40:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.
00001D50:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00001D60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 47 5F 13 80 7F ].@].@].@].G_.€
00001D70:46 DE DA C9 EF E6 DD EE E5 DD F0 E8 E0 F1 E8 E0 FÞÚÉïæÝîåÝðèàñèà
00001D80:E2 D2 C3 DE CD BD E3 D3 C3 E3 D4 C5 E2 D3 C5 E6 âÒÃÞÍ½ãÓÃãÔÅâÓÅæ
00001D90:D9 CA E5 D7 C9 E7 DA CB E9 DA CC E5 D6 C8 E2 D3 ÙÊå×ÉçÚËéÚÌåÖÈâÓ
00001DA0:C4 E0 D1 C2 DA CA B9 D3 C2 B1 CF BD AA CC B9 A6 ÄàÑÂÚÊ¹ÓÂ±Ï½ªÌ¹¦
00001DB0:D6 C7 B8 E1 D5 C9 E7 DB CF DF D3 C6 E0 D3 C7 DE ÖÇ¸áÕÉçÛÏßÓÆàÓÇÞ
00001DC0:D1 C5 D4 C9 B9 7F 85 5B 75 83 4A 74 78 3A 4A 5F ÑÅÔÉ¹…[uƒJtx:J_
00001DD0:14 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .A\.C\.C\.C\.C\.
00001DE0:42 5B 0B 42 5B 0C 42 5B 0C 43 5B 0C 42 5B 0B 42 B[.B[.B[.C[.B[.B
00001DF0:5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B 42 5B [.B[.B[.CZ.CZ.B[
00001E00:0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A .B[.DZ.C\.C\.C].
00001E10:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 C].C].C\.C\.C\.C
00001E20:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\
00001E30:0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 5E 0E 3F 5F .B[.D[..@].A^.?_
00001E40:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E 0E 40 5D 0D .?_.?_.?_.A^.@].
00001E50:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 @].@].@].A^.@].@
00001E60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
00001E70:0D 40 5D 0D 40 5D 0D 4B 61 15 8E 8B 57 CF CE B6 .@].@].KA.Ž‹WÏÎ¶
00001E80:EA E2 D6 F0 E7 DF F1 E9 E1 F1 E8 E0 E4 D5 C5 E1 êâÖðçßñéáñèàäÕÅá
00001E90:D1 C0 E0 D0 C0 E0 D0 C2 E4 D5 C6 E3 D4 C5 E3 D4 ÑÀàÐÀàÐÂäÕÆãÔÅãÔ
00001EA0:C5 E5 D6 C8 E8 DB CD EB DD D0 EB DD CF E8 DA CC ÅåÖÈèÛÍëÝÐëÝÏèÚÌ
00001EB0:E0 D1 C2 D7 C6 B5 CE BB A9 C9 B5 A2 D3 C4 B5 D3 àÑÂ×ÆµÎ»©Éµ¢ÓÄµÓ
00001EC0:C4 B6 DA CD BF E1 D5 C9 E3 D7 CB DD D1 C4 EA DF Ä¶ÚÍ¿áÕÉã×ËÝÑÄêß
00001ED0:D4 B7 B4 9C 77 85 4D 76 7D 3E 4F 61 17 41 5C 0C Ô•´œw…Mv}>Oa.A\.
00001EE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 42 C\.C\.C\.C\.B[.B
00001EF0:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00001F00:0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B .B[.B[.B[.CZ.CZ.
00001F10:44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 DZ.C\.C\.C].C].C
00001F20:5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C ].C\.C\.C\.C\.C\
00001F30:0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5A 0B .C\.C\.C\.B[.CZ.
00001F40:44 5B 09 00 40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E D[..@].@].?_.?_.
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 ].@].A^.@].@].@]
00001F70:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
00001F80:40 5D 0D 50 63 1A A3 9F 71 CA C9 B1 E0 D9 CB F1 @].Pc.£ŸqÊÉ±àÙËñ
00001F90:E8 E0 F2 EA E2 F2 E9 E0 E7 D8 C8 E4 D4 C4 E1 D0 èàòêâòéàçØÈäÔÄáÐ
00001FA0:C1 E4 D5 C6 DD CD BD D7 C6 B6 DE CF BF E5 D6 C8 ÁäÕÆÝÍ½×Æ¶ÞÏ¿åÖÈ
00001FB0:EB DE D0 F0 E3 D6 F1 E5 D8 F0 E3 D6 ED DF D3 E9 ëÞÐðãÖñåØðãÖíßÓé
00001FC0:DB CD E8 DA CC E3 D4 C5 E3 D7 CA E4 D8 CC D9 CB ÛÍèÚÌãÔÅã×ÊäØÌÙË
00001FD0:BD D9 CB BD E0 D3 C7 DF D3 C6 E7 DC D0 E1 D8 C8 ½ÙË½àÓÇßÓÆçÜÐáØÈ
00001FE0:80 8D 56 7A 82 45 55 64 1B 41 5C 0C 43 5C 0C 43 €□Vz‚EUd.A\.C\.C
00001FF0:5C 0C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B \.C\.B[.B[.B[.B[
00002000:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B .B[.B[.B[.B[.B[.
00002010:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 B[.B[.DZ.DZ.DZ.C
00002020:5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C \.C\.C].C].C].C\
00002030:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
00002040:43 5C 0C 43 5C 0C 43 5A 0B 44 5A 0B 44 5B 09 00 C\.C\.CZ.DZ.D[..
00002050:40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F @].@].?_.?_.?_.?
00002060:5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D _.@].@].@].@].@]
00002070:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .A^.@].@].@].@].
00002080:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 58 @].@].@].@].@].X
00002090:67 20 B2 AF 88 D8 D4 C5 F0 E7 DE F1 E9 E1 F3 EC g ²¯ˆØÔÅðçÞñéáóì
000020A0:E4 F3 E9 E1 E9 DA CB E6 D6 C7 E4 D4 C4 E6 D8 C9 äóéáéÚËæÖÇäÔÄæØÉ
000020B0:DF D0 C0 D5 C4 B3 DB CB BA E5 D7 C9 EB DD D0 EF ßÐÀÕÄ³ÛË°å×ÉëÝÐï
000020C0:E2 D5 F1 E4 D8 F0 E4 D7 ED DF D2 EE E1 D4 E6 D8 âÕñäØðä×íßÒîáÔæØ
000020D0:C9 C1 AE 98 CA B9 A8 E8 DC D1 EC E1 D7 E7 DC D0 ÉÁ®˜Ê¹¨èÜÑìá×çÜÐ
000020E0:DD D0 C3 D4 C6 B7 D8 CB BC AE AC 8B 79 87 4D 7B ÝÐÃÔÆ·ØË¼®¬‹y‡M{
000020F0:84 4A 5C 67 20 41 5C 0C 43 5C 0C 43 5C 0C 43 5C „J\g A\.C\.C\.C\
```

```
00002100:0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.
00002110:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002120:5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C [.DZ.DZ.DZ.C\.C\
00002130:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C .C].C].C].C\.C\.
00002140:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 C\.C\.C\.C\.C\.B
00002150:5B 0B 44 5A 0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 [.DZ.DZ.D[..@].@
00002160:5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D ].?_.?_.?_.?_.@]
00002170:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.
00002180:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00002190:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 64 6C 29 C3 BD ].@].@].@].Dl)Ã½
000021A0:A0 F4 EC E3 F2 E9 E0 F3 EA E2 F5 EE E6 F3 EA E1 ôìãòéàóêâõîæóêá
000021B0:EB DC CD E8 D9 CA E5 D5 C6 E4 D4 C5 E3 D3 C3 D9 ëÜÍèÙÊåÕÆäÔÅãÓÃÙ
000021C0:C8 B8 D3 C1 AF DC CC BC EB DD D0 EF E2 D6 F2 E5 È¸ÓÁ¯ÜÌ¼ëÝÐïâÖòå
000021D0:DA F2 E5 D9 F4 E7 DB EA DC CE C7 B3 A0 B7 A2 8B ÚòåÙôçۊÜÎÇ³ •¢‹
000021E0:D1 C2 B3 EB E0 D6 E6 DA CF EA E0 D5 EF E4 DB E8 ÑÂ³ëàÖæÚÏêàÕïäÛè
000021F0:DC D1 D0 C5 B4 8A 8F 63 7C 8B 51 79 85 4A 64 69 ÜÑÐÅ´Š□C|‹Qy…Jdi
00002200:27 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 'A\.C\.C\.C\.B[.
00002210:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002220:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A [.B[.B[.B[.B[.DZ
00002230:0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A .DZ.DZ.C\.C\.C].
00002240:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 C].C].C\.C\.C\.C
00002250:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 44 5A 0B 44 5A \.C\.C\.B[.DZ.DZ
00002260:0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 5D 0D 3F 5E .DZ.D[..@].@].?^
00002270:0D 3F 5E 0D 3F 5E 0D 3F 5E 0D 40 5D 0D 40 5D 0D .?^.?^.?^.@].@].
00002280:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00002290:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000022A0:0D 3F 5C 0C 40 5C 0D 72 74 35 CC C5 AC F4 EC E4 .?\.@\.rt5ÌÅ¬ôìä
000022B0:F1 E9 E0 F6 EE E6 F5 EE E6 F2 E9 DE ED DE CF EB ñéàöîæõîæòéÞíÞÏë
000022C0:DC CE E3 D3 C3 D8 C7 B4 DB CB BA DF D0 BF D9 C8 ÜÎãÓÃØÇ´ÛË°ßÐ¿ÙÈ
000022D0:B7 D9 C8 B7 ED DE D0 F4 E6 D9 F9 ED E0 FD F0 E4 •ÙÈ•íÞÐôæÙùíàýðä
000022E0:FF F3 E7 E0 CF BD CA B6 A1 C7 B3 9D F3 E7 DC F4 ÿóçàϽʶ¡Ç³•óçÜô
000022F0:EA E0 E7 DD D0 E7 DC D0 E8 DD D2 EF E5 DB F0 E7 êàçÝÐçÜÐèÝÒïåÛðç
00002300:DD A5 A6 81 7C 8A 50 76 83 47 6F 71 31 43 5C 0E Ý¥¦□|ŠPvƒGoq1C\.
00002310:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 C\.B[.B[.B[.B[.B
00002320:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00002330:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B .B[.B[.B[.DZ.DZ.
00002340:43 5A 0B 42 5B 0B 42 5B 0B 43 5C 0A 43 5C 0A 43 CZ.B[.B[.C\.C\.C
00002350:5C 0A 43 5C 0C 43 5C 0C 43 5C 0B 43 5C 0B 43 5C \.C\.C\.C\.C\.C\
00002360:0C 43 5B 0C 42 5A 0A 43 59 0A 44 5A 0B 44 5A 0B .C[.BZ.CY.DZ.DZ.
00002370:43 5B 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D C[..@].@].@].@].
00002380:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00002390:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000023A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C .@].@].@].@].?\.
000023B0:43 5E 0F 80 7E 41 D3 CD BA F2 E9 E0 F4 EB E2 F4 C^.€~AÓÍ°òéàôëâô
000023C0:EA E1 F0 E8 DF E5 D9 CD DD CE BE E4 D5 C7 DC CD êáðèßåÙÍÝÎ¾äÕÇÜÍ
000023D0:BF D9 CA BB D9 CA BB DA CC BD D2 C4 B6 CC BF B2 ¿ÙÊ»ÙÊ»ÚÌ½ÒÄ¶Ì¿²
000023E0:CC C1 B5 C8 BD B2 C3 B9 AF C0 B6 AB BA AE A2 AA ÌÁµÈ½²Ã¹¯À¶«°®¢ª
000023F0:9A 8B A6 95 84 BB AD 9E D7 CE C5 D5 CA BF D3 C6 š‹¦•„»-ž×ÎÅÕÊ¿ÓÆ
00002400:B9 DC CF C1 E5 D7 C9 EB DF D2 E6 DC CF 8D 92 67 ¹ÜÏÁå×ÉëßÒæÜÏ•'g
00002410:7A 87 4D 73 82 45 79 79 39 49 5E 11 43 5C 0C 42 z‡Ms‚Eyy9I^.C\.B
00002420:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00002430:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.
00002440:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 B[.B[.DZ.DZ.B[.B
00002450:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C [.B[.C\.C\.C\.C\
00002460:0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B .C\.C].C].C\.DZ.
00002470:43 59 0A 43 59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 CY.CY.DZ.DZ.B\..
00002480:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
00002490:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000024A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000024B0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 45 5F 11 90 @].@].@].?\.E_.•
000024C0:8A 4F EA E1 D5 E8 DF D7 E7 DF D6 E7 E1 DA E5 E0 ŠOêáÕèß×çßÖçáÚåà
000024D0:DB E6 E3 DF EA E8 E5 F0 EE ED F2 F2 F1 F1 F1 F1 Ûæãßêèåðîíòòñññ
000024E0:F0 F0 F0 EB EB EB E3 E4 E4 D7 D7 D8 C2 C3 C4 A8 ðððëëëãää××ØÂÃÄ¨
000024F0:A9 AA 8B 8B 8C 6C 6C 6D 53 54 55 45 45 46 3E 3F ©ª‹‹ŒllmSTUEEF>?
00002500:3F 3B 3B 3B 3C 3B 3A 41 40 3F 4A 48 46 5B 57 54 ?;;;<;:A@?JHF[WT
00002510:78 72 6C 9C 94 8B B4 AB A2 A2 9F 86 95 9C 6F 7A xrlœ"‹´«¢¢Ÿ†•œoz
00002520:8A 4F 7B 7D 3D 4B 5F 13 43 5C 0C 42 5B 0B 42 5B ŠO{}=K_.C\.B[.B[
00002530:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.
00002540:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002550:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B [.DZ.DZ.B[.B[.B[
```

```
00002560:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.C\.
00002570:43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B 43 59 0A 43 C].C].C\.DZ.CY.C
00002580:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0D 40 Y.DZ.DZ.B\..@].@
00002590:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000025A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000025B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000025C0:5D 0D 40 5D 0D 3F 5C 0C 52 64 1D A1 92 80 C5 C5 ].@].?\.Rd.¡'€ÅÅ
000025D0:C4 D2 D2 D2 E1 E2 E3 EB EC ED F2 F2 F3 F5 F5 F5 ÄÒÒÒáâãëìíòòóõõõ
000025E0:F7 F7 F6 F6 F6 F6 F5 F4 F3 F4 F3 F1 F1 EF EE F1 ÷÷öööõõóóóóññïîñ
000025F0:EF ED ED EC EA E7 E5 E3 DF DE DC D2 D1 D0 C4 C2 ïííìêçåãßÞÜÒÑÐÄÂ
00002600:C0 B7 B5 B2 AB A9 A6 A0 9D 9B 93 90 8D 89 88 87 À•µ²«©¦ □›"□□‰^‡
00002610:7E 7E 7F 72 72 71 6B 6A 6A 69 69 6A 69 6A 6B 6F ~~rrqkjjiijijko
00002620:71 73 83 86 89 A2 A3 A5 BB BA BA C4 C7 BF AE A4 qsƒ†‰¢£¥»°°ÄÇ¿®¤
00002630:81 55 64 1C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B •UD.C\.B[.B[.B[.
00002640:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002650:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A [.B[.B[.B[.B[.DZ
00002660:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C .DZ.B[.B[.B[.C\.
00002670:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43 C\.C\.C\.C\.C\.C].C
00002680:5D 0A 43 5C 0C 44 5A 0B 44 5A 0B 43 59 0A 43 59 ].C\.DZ.DZ.CY.CY
00002690:0A 44 5A 0B 42 5C 09 00 40 5D 0D 40 5D 0D 40 5D .DZ.B\..@].@].@]
000026A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000026B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000026C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000026D0:0D 3F 5C 0C 75 82 5E 9E A0 A2 C6 C7 C8 DE DE DE .?\.u‚^ž ¢ÆÇÈÞÞÞ
000026E0:EA E9 E8 F2 F1 F0 F5 F3 F0 F5 F3 F0 F1 ED E9 EB êéèòñðõóðõóðñíéë
000026F0:E6 E0 E9 E3 DD E6 E1 DA E0 DB D6 E9 E5 DD F6 F4 æàéãÝæáÚàÛÖéåÝöô
00002700:F3 E9 E5 E1 E4 DF DA DC D7 D0 D2 CA C1 D4 CC C4 óéåáäßÚÜ×ÐÒÊÁÔÌÄ
00002710:DB D4 CC E2 DC D5 D4 CC C3 CF C7 BC F3 EF EB DF ÛÔÌâÜÕÔÌÃÏÇ¼óïëß
00002720:D9 D2 D9 D2 CB DC D7 D1 DB D8 D4 C9 C4 B8 B8 AF ÙÒÙÒËÜ×ÑÛØÔÉÄ¸¸¯
00002730:96 C3 BF B7 D0 D0 D0 D0 D1 D2 CF D1 D3 86 8F 6D –Ã¿•ÐÐÐÐÑÒÏÑÓ†•m
00002740:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 C\.B[.B[.B[.B[.B
00002750:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00002760:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B .B[.B[.B[.DZ.DZ.
00002770:42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 B[.B[.B[.C\.C\.C
00002780:5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A \.C\.C\.C].C].DZ
00002790:0B 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 44 5A 0B .DZ.DZ.CY.CY.DZ.
000027A0:42 5C 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D B\..@].@].@].@].
000027B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000027C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000027D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C .@].@].@].@].?\.
000027E0:89 91 7B AD AC AC AD A4 84 A2 97 5E A7 9B 5F EE ‰'{-¬¬-¤„¢--^§›_î
000027F0:E9 DA FC FC FC F8 F6 F4 EE E9 E4 E5 DE D7 E6 E1 éÚüüüøöôîéäåÞ×æá
00002800:DB E4 E0 DB B8 AA 7D B7 AD 7C FF FF FF F2 F2 F2 ÛäàÛ¸ª}•-|ÿÿÿòòò
00002810:E5 E1 DD DE D9 D3 D6 CF C8 DB D6 D0 DB D6 D0 D9 åáÝÞÙÓÖÏÈÛÖÐÛÖÐÙ
00002820:D3 CD D1 CA C2 BE B4 A8 BA AF A3 C6 BD B3 CE C6 ÓÍÑÊÂ¾´¨º¯£Æ½³ÎÆ
00002830:BE D1 CD C7 C5 B8 99 9A 8F 4D 91 88 43 9A 90 4E ¾ÑÍÇÅ¸™šM'ˆCšN
00002840:A5 99 61 B2 A9 8B CB CC 9D A4 90 43 5C 0C 42 ¥™a²©‹ËËÌ•¤•C\.B
00002850:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00002860:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.
00002870:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 B[.B[.DZ.DZ.B[.B
00002880:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C [.B[.C\.C\.C\.C\
00002890:0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A 0B 44 5A 0B .C\.C].C].DZ.DZ.
000028A0:43 59 0A 43 59 0A 43 59 0A 44 5A 0B 42 5C 09 00 CY.CY.CY.DZ.B\..
000028B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000028C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@]
000028D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000028E0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 90 99 7F 7D @].@].@].?\.□™}
000028F0:7C 7C 73 70 69 83 7D 64 93 88 5F A6 9C 78 EA E8 ||spiƒ}D"ˆ_¦œxêè
00002900:E4 FB F9 F7 E7 E1 DA E1 D9 D1 D9 D0 BF AB 9F 68 äûù÷çáÚáÙÑÙÐ¿«Ÿh
00002910:8D 84 3E 99 8F 4C C8 BB 90 E6 E0 CA F3 F0 EB E9 □„>™□LÈ»□æàÊóðëé
00002920:E5 E1 E0 DA D4 E6 E2 DE DD D7 D2 CE C7 BF DF DA åáàÚÔæâÞÝ×ÒÎÇ¿ßÚ
00002930:D5 E1 DC D7 D5 CE C6 D6 CF C7 CC C3 B4 BC B1 99 ÕáÜ×ÕÎÆÖÏÇÌÃ´¼±™
00002940:96 8A 54 8C 80 47 90 84 56 8E 83 5D 89 82 6C 9A –ŠTŒ€G□„VŽƒ]‰‚lš
00002950:97 91 B2 B2 B3 92 9B 80 43 5C 0C 42 5B 0B 42 5B --'²²³'›€C\.B[.B[
00002960:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.
00002970:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002980:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B [.DZ.DZ.B[.B[.B[
00002990:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\.
000029A0:43 5D 0A 42 5C 09 44 5A 0B 44 5A 0B 43 59 0A 43 C].B\.DZ.DZ.CY.C
000029B0:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0B 40 Y.DZ.DZ.B\..@].@
```

```
000029C0:5D 0B 40 5E 0C 40 5E 0C 40 5E 0C 40 5E 0C 40 5E ].@^.@^.@^.@^.@^
000029D0:0C 40 5D 0B 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].
000029E0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@
000029F0:5D 0D 40 5D 0D 3F 5C 0C 49 63 1A 85 94 6B A3 A7 ].@].?\.IC.…"k£§
00002A00:9E 9F A1 A3 9E A1 A5 A3 A4 A6 B9 B8 B7 EA E7 E5 žŸ¡£ž¡¥£¤¦¹¸•êçå
00002A10:DF D6 CD CE C8 C2 B6 B0 A3 A8 9F 81 A7 9F 81 A2 ßÖÍÎÈÂ¶°£¨Ÿ□§Ÿ□¢
00002A20:98 77 9A 8F 66 9A 91 6B A2 9B 84 B5 B4 AB D4 D1 ˜wš□Fš'k¢›„µ´«ÔÑ
00002A30:CD DE D9 D3 DE D8 D1 D0 C9 C1 D0 CA C4 DF DD DC ÍÞÙÓÞØÑÐÉÁÐÊÄßÝÜ
00002A40:E4 E1 DE CB C8 C6 7E 7A 71 57 52 49 56 53 50 5B äáÞËÈÆ~zqWRIVSP[
00002A50:5C 5E 6E 71 74 86 88 8B A0 A1 A3 A6 AB A0 82 8F \^nqt†ˆ‹ ¡£¦« ‚•
00002A60:66 4C 63 18 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B FLC.B[.B[.B[.B[.
00002A70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002A80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A [.B[.B[.B[.B[.DZ
00002A90:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C .DZ.B[.B[.B[.C\.
00002AA0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 09 43 C\.C\.C\.C\.C\.C[.C
00002AB0:5B 09 44 5A 09 44 5A 0B 43 59 0A 43 59 0A 43 59 [.DZ.DZ.CY.CY.CY
00002AC0:0A 43 59 0A 43 5A 0A 00 40 5D 0D 40 5E 0E 41 5E .CY.CZ..@].@^.A^
00002AD0:0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D .A^.A^.A^.@].@].
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 .@].?\.?\.D`.Xp.
00002B10:73 85 53 84 92 6B 8F 99 7C A6 AC 9A B7 B8 AE B9 s…S„'k□™|¦¬š•¸®¹
00002B20:B9 B8 BB BD BE BA BD BF B8 BB BD B5 B8 BB B0 B3 ¹¸»½¾º½¿¸»½µ¸»°³
00002B30:B6 A9 AB AD 9D 9E 9F 8F 8F 90 86 86 86 87 86 86 ¶©«-□žŸ□□□†††‡††
00002B40:B5 B1 AD DE D8 D1 BB B5 AF 6C 6C 6C 79 79 79 6F µ±-ÞØÑ»µ¯lllyyyo
00002B50:6F 6F 71 72 71 7A 7C 76 7D 83 73 7C 86 68 7B 86 ooqrqz|v}ƒs|†h{†
00002B60:60 6F 7E 4D 57 6B 2A 46 5E 11 43 5C 0C 43 5D 0A `o~MWk*F^.C\.C].
00002B70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 B[.B[.B[.B[.B[.B
00002B80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[
00002B90:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5A 0B 42 5A 0B .B[.B[.B[.BZ.BZ.
00002BA0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 B[.B[.B[.B[.C\.C
00002BB0:5C 0A 43 5C 0A 43 5C 0A 44 5B 09 44 5A 0A 44 5A \.C\.C\.D[.DZ.DZ
00002BC0:0A 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 59 0A .DZ.DZ.CY.CY.CY.
00002BD0:44 5A 0B 00 3E 5E 0D 3F 5D 0D 3F 5D 0D 40 5E 0E DZ..>^.?].?].@^.
00002BE0:41 5E 0E 40 5D 0D 40 5D 0D 3F 5C 0C 40 5D 0C 40 A^.@].@].?\.@].@
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 ].@].A\.HB.Ri#Xn
00002C30:2D 5D 72 34 61 75 3A 64 77 3F 67 79 42 68 7A 44 -]r4Au:Dw?gyBhzD
00002C40:69 7B 46 69 7A 46 68 78 44 66 77 42 64 75 40 8C i{FizFhxDFwBDu@Œ
00002C50:96 6D C0 BE B0 51 62 2B 57 6B 2D 58 6D 2C 51 67 –mÀ¾°QB+Wk-Xm,Qg
00002C60:22 49 60 15 41 5B 0B 41 5B 0B 41 5B 0B 41 5B 0B "I`.A[.A[.A[.A[.
00002C70:41 5B 0B 40 5B 0B 40 5B 0B 40 5C 09 42 5B 0A 42 A[.@[.@[.@\.B[.B
00002C80:5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B [.B[.B[.B[.B[.B[
00002C90:0A 41 5A 0A 42 5A 0C 42 5A 0C 42 5A 0C 42 5B 0B .AZ.BZ.BZ.BZ.B[.
00002CA0:42 5B 0B 42 5B 0B 42 5B 0A 42 5B 0A 44 5A 0B 44 B[.B[.B[.B[.DZ.D
00002CB0:5A 0B 44 5A 0A 44 5A 0A 44 5A 0A 45 5B 0B 45 5B Z.DZ.DZ.DZ.E[.E[
00002CC0:0B 45 5B 0B 44 5A 09 44 5A 09 44 5A 0B 44 5A 0B .E[.DZ.DZ.DZ.DZ.
00002CD0:44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 5B 09 00 DZ.DZ.CY.CY.C[..
00002CE0:3F 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F 0D 40 5E 0D 40 ?_.?_.?_.?_.@^.@
00002CF0:5E 0D 40 5E 0D 40 5E 0D 40 5F 0C 40 5F 0C 40 5F ^.@^.@^.@_.@_.@_
00002D00:0C 40 5F 0C 3F 5E 0B 3F 5E 0B 40 5E 0B 40 5E 0B .@_.?^.?^.@^.@^.
00002D10:40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 @].@].@].@].@].@
00002D20:5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D ].@].@].@].@].@]
00002D30:0C 41 5D 0C 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B .A].A\.A\.A\.A\.
00002D40:41 5D 0C 41 5D 0C 41 5D 0C 40 5C 0B 40 5C 0B 40 A].A].A].@\.@\.@
00002D50:5C 0B 40 5C 0B 40 5C 0C 41 5C 0C 41 5C 0C 5A 70 \.@\.@\.A\.A\.Zp
00002D60:2B 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C +A\.A\.A\.A\.A\.
00002D70:41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41 A\.A\.A\.A\.A\.A
00002D80:5C 0B 41 5C 0B 41 5D 0B 42 5D 0A 42 5D 0A 42 5D \.A\.A].B].B].B]
00002D90:0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A .B].B].B].B].B].
00002DA0:42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 43 B\.B\.B\.B\.B\.C
00002DB0:5C 0B 43 5D 0A 43 5D 0A 44 5C 0B 44 5C 0B 44 5C \.C].C].D\.D\.D\
00002DC0:0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A .D\.D\.D\.D\.D\.
00002DD0:43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43 C[.C[.C[.C[.C[.C
00002DE0:5B 0A 43 5B 0A 43 5B 0A 43 5D 09 00 3F 61 0C 40 [.C[.C[.C]..?A.@
00002DF0:62 0D 40 62 0D 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 B.@B.?A.?A.?A.?A
00002E00:0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C .?A.?A.?A.?A.?A.
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 .@`.?_.?_.?_.A_.
00002E40:41 5E 0B 41 5E 0B 42 5F 0C 42 5F 0C 40 60 0C 40 A^.A^.B_.B_.@`.@
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 .?_.B_.B_.A^.A^.
00002E70:41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0C 41 5E 0B 41 A^.A^.A^.A_.A^.A
00002E80:5E 0B 41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0B 41 5F ^.A^.A^.A^.A_.A_
00002E90:0B 41 5F 0A 41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B .A_.A_.A_.A_.A_.
00002EA0:41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 42 5F 0B 42 A_.A_.A_.A_.B_.B
00002EB0:5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 43 5F 0B 43 5E _.A_.A_.A_.C_.C^
00002EC0:0A 43 5E 0A 43 5F 0B 43 5F 0B 43 5F 0B 43 5F 0B .C^.C_.C_.C_.C_.
00002ED0:43 5F 0B 43 5F 0A 43 5F 0A 43 5F 0A 42 5E 09 42 C_.C_.C_.C_.B^.B
00002EE0:5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E ^.B^.B^.B^.B^.B^
00002EF0:09 42 5E 09 42 5E 09 00 3F 64 0B 3F 64 0B 3F 64 .B^.B^..?D.?D.?D
00002F00:0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B .?D.?D.?D.?D.?D.
00002F10:3F 64 0C 3F 64 0C 3F 64 0C 3F 62 0C 40 62 0C 40 ?D.?D.?D.?B.@B.@
00002F20:62 0C 40 62 0C 40 62 0C 40 62 0C 40 62 0C 3F 62 B.@B.@B.@B.@B.?B
00002F30:0C 3F 62 0C 40 63 0C 40 63 0D 40 62 0C 40 62 0C .?B.@C.@C.@B.@B.
00002F40:3F 62 0B 3F 62 0B 3F 62 0C 40 62 0C 40 62 0C 41 ?B.?B.?B.@B.@B.A
00002F50:62 0C 41 63 0D 40 62 0C 40 63 0A 40 63 0A 40 63 B.AC.@B.@C.@C.@C
00002F60:0A 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C .?B.?B.?B.?B.?B.
00002F70:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 @A.@A.@A.@A.@A.@
00002F80:61 0B 40 61 0B 41 62 0C 40 61 0B 40 61 0B 40 61 A.@A.AB.@A.@A.@A
00002F90:0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B .@A.@A.@A.@A.@A.
00002FA0:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 @A.@A.@A.@A.@A.@
00002FB0:61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 A.@A.@A.@A.@A.@A
00002FC0:0B 40 62 09 40 62 09 40 62 09 42 61 09 42 61 09 .@B.@B.@B.BA.BA.
00002FD0:42 61 09 42 61 09 42 61 09 42 61 09 42 61 09 42 BA.BA.BA.BA.BA.B
00002FE0:61 09 42 61 09 42 61 09 40 62 09 42 61 09 42 61 A.BA.BA.@B.BA.BA
00002FF0:09 42 61 09 42 61 09 42 61 09 42 61 09 42 61 09 .BA.BA.BA.BA.BA.
00003000:42 61 09 00 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F BA..=`.=`.=`.=`.
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 .=`.=A.=A.<`.<`.
00003040:3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D =A.=A.=A.=A.=A.=
00003050:61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 A.=A.=A.=A.=A.=A
00003060:0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E .=A.=A.=A.=A.=A.
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 ?`.?`.?_.?_.=_..
00003110:12 00 00 00 0C 00 00 00 01 00 00 00 52 00 00 00 ............R...
00003120:70 01 00 00 02 00 00 00 F3 FF FF FF 00 00 00 00 p.......óÿÿÿ....
00003130:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 01 N...N...È.......
00003140:04 00 00 02 41 00 72 00 69 00 61 00 6C 00 00 00 ....A.r.i.A.l...
00003150:00 00 00 00 00 00 00 00 00 00 00 00 0C 45 00 00 ............E..
00003160:BC 16 E8 FE FE 07 00 00 20 00 CC 00 00 00 00 00 ¼.èþþ... .Ì.....
00003170:4C 00 00 00 FE 07 00 00 DA 16 01 B8 FF FF FF FF L...þ...Ú..¸ÿÿÿÿ
00003180:00 00 00 00 00 00 00 00 F0 F6 13 00 00 00 00 00 ........ðö......
00003190:0E 20 05 27 00 00 00 00 28 00 00 00 00 00 00 00 . .'....(.......
000031A0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ./..............
000031B0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000031C0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000031D0:00 00 00 00 00 00 00 00 28 00 00 00 FF FF FF 00 ........(...ÿÿÿ.
000031E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ./..............
000031F0:58 00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00 X...,...........
00003200:00 00 80 3F 00 00 00 00 00 00 00 00 00 00 80 3F ..€?..........€?
00003210:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00003220:D0 C5 35 00 00 00 00 00 28 00 00 00 59 00 00 00 ÐÅ5.....(...Y...
00003230:2D 00 00 00 01 00 18 00 00 00 00 00 1C 2F 00 00 -............/..
00003240:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00003250:10 00 90 01 00 00 00 00 25 00 00 00 00 00 00 00 ..•.....%.......
00003260:D3 3F EC FE FE 07 00 00 79 0D 21 11 00 00 00 00 Ó?ìþþ...y.!.....
00003270:10 00 90 01 00 00 00 00 00 00 00 00 59 00 00 00 ..•.........Y...
```

```
00003280:2D 00 00 00 64 76 00 08 00 00 00 00 25 00 00 00 -...Dv......%...
00003290:0C 00 00 00 02 00 00 00 54 00 00 00 A0 00 00 00 ........T... ...
000032A0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF ........ÿÿÿÿÿÿÿÿ
000032B0:01 00 00 00 AB 0A 0D 42 00 00 0D 42 12 00 00 00 ....«..B...B....
000032C0:05 00 00 00 0E 00 00 00 4C 00 00 00 00 00 00 00 ........L.......
000032D0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF ........ÿÿÿÿÿÿÿÿ
000032E0:68 00 00 00 53 00 69 00 6D 00 70 00 6C 00 65 00 h...S.i.m.p.l.E.
000032F0:20 00 53 00 61 00 6D 00 70 00 6C 00 65 00 00 00 .S.A.m.p.l.E...
00003300:09 00 00 00 03 00 00 00 0B 00 00 00 07 00 00 00 ................
00003310:03 00 00 00 07 00 00 00 04 00 00 00 09 00 00 00 ................
00003320:07 00 00 00 0B 00 00 00 07 00 00 00 03 00 00 00 ................
00003330:07 00 00 00 09 00 00 00 52 00 00 00 70 01 00 00 ........R...p...
00003340:03 00 00 00 F3 FF FF FF 00 00 00 00 4E 0C 00 00 ....óÿÿÿ....N...
00003350:4E 0C 00 00 C8 00 00 00 00 00 00 00 04 00 00 02 N...È...........
00003360:4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 66 00 M.i.C.r.o.s.o.F.
00003370:74 00 20 00 53 00 61 00 6E 00 73 00 20 00 53 00 t. .S.A.n.s. .S.
00003380:65 00 72 00 69 00 66 00 00 00 00 00 00 00 00 00 E.r.i.F.........
00003390:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000033A0:00 00 00 00 28 00 00 00 59 00 00 00 2D 00 00 00 ....(...Y...-...
000033B0:01 00 18 00 00 00 00 00 1C 2F 00 00 00 00 00 00 ........./......
000033C0:00 00 00 00 00 00 00 00 00 00 00 00 10 00 90 01 ..............•.
000033D0:00 00 00 00 25 00 00 00 00 00 00 00 D3 3F EC FE ....%.......Ó?ìþ
000033E0:FE 07 00 00 79 0D 21 11 00 00 00 00 10 00 90 01 þ...y.!.......•.
000033F0:00 00 00 00 00 00 00 00 59 00 00 00 2D 00 00 00 ........Y...-...
00003400:64 76 00 08 00 00 00 00 00 00 00 00 00 00 08 0D 00 Dv..............
00003410:0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00003420:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00003430:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00003440:00 00 00 00 92 A0 CD 02 00 00 00 00 CA BE CD 02 ....' Í.....Ê¾Í.
00003450:00 00 00 00 00 00 00 00 00 00 00 00 FF FF 5A FE ............ÿÿZþ
00003460:00 00 00 00 00 00 00 00 00 00 00 00 95 F1 53 FE ............•ñSþ
00003470:FE 07 00 00 BE 06 5A FE FE 07 00 00 87 F2 53 FE þ...¾.Zþþ...‡òSþ
00003480:FE 07 00 00 C4 04 5A FE FE 07 00 00 79 0D 21 11 þ...Ä.Zþþ...y.!.
00003490:00 00 00 00 01 00 00 00 00 00 00 00 F0 02 5A FE ............ð.Zþ
000034A0:64 76 00 08 00 00 00 00 25 00 00 00 0C 00 00 00 Dv......%.......
000034B0:03 00 00 00 52 00 00 00 70 01 00 00 04 00 00 00 ....R...p.......
000034C0:F2 FF FF FF 00 00 00 00 4E 0C 00 00 4E 0C 00 00 òÿÿÿ....N...N...
000034D0:C8 00 00 00 00 00 00 00 04 00 00 02 4D 00 69 00 È...........M.i.
000034E0:63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 20 00 C.r.o.s.o.F.t. .
000034F0:53 00 61 00 6E 00 73 00 20 00 53 00 65 00 72 00 S.A.n.s. .S.E.r.
00003500:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00 i.F.............
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 Èñ........•.....
00003530:20 AC CF 02 00 00 00 00 10 00 00 00 06 00 00 00 ¬Ï.............
00003540:06 00 00 00 04 00 00 00 01 00 00 00 01 00 00 00 ................
00003550:01 00 00 00 01 00 00 00 0D 00 00 00 00 00 00 00 ................
00003560:03 00 00 00 00 00 08 00 00 3B 09 00 00 00 00 00 ........;.......
00003570:A0 E8 07 02 00 00 00 00 03 00 00 00 FE 07 00 00 è..........þ...
00003580:90 01 00 00 4D 00 69 00 63 00 72 00 6F 00 73 00 •...M.i.C.r.o.s.
00003590:6F 00 66 00 74 00 20 00 53 00 61 00 00 00 73 00 o.F.t. .S.A...s.
000035A0:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00 .S.E.r.i.F.....
000035B0:00 00 00 00 00 00 00 00 FF FF 5A FE 00 00 00 00 ........ÿÿZþ....
000035C0:40 02 5A FE FE 07 00 00 9D 04 00 00 00 00 00 00 @.Zþþ...•.......
000035D0:FF FF FF FF FF FF FF FF 01 00 00 00 00 00 00 00 ÿÿÿÿÿÿÿÿ........
000035E0:20 AC CF 02 00 00 00 00 00 00 07 02 00 00 00 00 ¬Ï.............
000035F0:10 AC CF 02 00 00 00 00 26 36 E3 76 00 00 00 00 .¬Ï.....&6ãv....
00003600:00 00 07 02 00 00 00 00 01 00 00 00 00 27 00 00 ............'..
00003610:00 00 00 00 00 00 00 00 20 AC CF 02 64 76 00 08 ........ ¬Ï.Dv..
00003620:00 00 00 00 25 00 00 00 0C 00 00 00 04 00 00 00 ....%...........
00003630:28 00 00 00 0C 00 00 00 03 00 00 00 52 00 00 00 (...........R...
00003640:70 01 00 00 03 00 00 00 13 00 00 00 00 00 00 00 p...............
00003650:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 00 N...N...È.......
00003660:04 00 00 02 4D 00 69 00 63 00 72 00 6F 00 73 00 ....M.i.C.r.o.s.
00003670:6F 00 66 00 74 00 20 00 53 00 61 00 6E 00 73 00 o.F.t. .S.A.n.s.
00003680:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00 .S.E.r.i.F.....
00003690:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000036A0:00 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 ÿÿZþ....@.Zþþ...
000036D0:9D 04 00 00 00 00 00 00 FF FF FF FF FF FF FF FF •.......ÿÿÿÿÿÿÿÿ
```

```
000036E0:01 00 00 00 00 00 00 00 20 AC CF 02 00 00 00 00 ........ ¬Ï.....
000036F0:00 00 07 02 00 00 00 00 40 02 5A FE FE 07 00 00 ........@.Zþþ...
00003700:F3 14 00 00 00 00 00 00 F3 14 0A 1E 00 00 00 00 ó.......ó.......
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00 "Šèþþ...........
00003720:65 58 53 FE 00 00 00 00 00 00 00 00 00 00 00 00 EXSþ............
00003730:00 F5 13 00 00 00 00 00 03 01 56 E5 89 1A 00 00 .õ........Vå‰...
00003740:55 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 U...............
00003750:00 00 00 00 FE 07 00 00 79 0D 21 11 00 00 00 00 ....þ...y.!.....
00003760:40 02 5A FE 00 00 00 00 26 06 5A FE FE 07 00 00 @.Zþ....&.Zþþ...
00003770:08 F5 13 00 00 00 00 00 00 F5 13 00 00 00 00 00 .õ.......õ......
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00 .ËTþþ...y.!.....
00003790:04 00 00 00 00 00 00 00 24 07 5A FE FE 07 00 00 ........$.Zþþ...
000037A0:01 00 00 00 64 76 00 08 00 00 00 00 25 00 00 00 ....Dv......%...
000037B0:0C 00 00 00 03 00 00 00 25 00 00 00 0C 00 00 00 ........%.......
000037C0:02 00 00 00 28 00 00 00 0C 00 00 00 04 00 00 00 ....(...........
000037D0:28 00 00 00 0C 00 00 00 03 00 00 00 25 00 00 00 (...........%...
000037E0:0C 00 00 00 0D 00 00 80 0E 00 00 00 14 00 00 00 .......€........
000037F0:00 00 00 00 10 00 00 00 14 00 00 00             ............
```

3.2.1 EMR_HEADER Example

This section provides an example of an 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
00000010:59 00 00 00 59 00 00 00 00 00 00 00 00 00 00 00
00000020:42 0C 00 00 41 0C 00 00 20 45 4D 46 00 00 01 00
00000030:FC 37 00 00 16 00 00 00 05 00 00 00 34 00 00 00
00000040:6C 00 00 00 00 00 00 00 80 07 00 00 B0 04 00 00
00000050:A5 02 00 00 A7 01 00 00 00 00 00 00 00 00 00 00
00000060:00 00 00 00 D5 55 0A 00 A5 75 06 00 53 00 61 00
00000070:6D 00 70 00 6C 00 65 00 20 00 45 00 4D 00 46 00
00000080:20 00 74 00 68 00 61 00 74 00 20 00 68 00 61 00
00000090:73 00 20 00 61 00 20 00 62 00 72 00 75 00 73 00
000000A0:68 00 20 00 66 00 69 00 6C 00 6C 00 2C 00 20 00
000000B0:62 00 69 00 74 00 6D 00 61 00 70 00 2C 00 20 00
000000C0:61 00 6E 00 64 00 20 00 74 00 65 00 78 00 74 00
000000D0:00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000001)																															
Size (0x000000D4)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0x00000059)																															
... (0x00000059)																															
Frame (0x00000000)																															
... (0x00000000)																															

... (0x00000C42)
... (0x00000C31)

Type (4 bytes): 0x00000001 identifies the record type as EMR_HEADER.

Size (4 bytes): 0x000000D4 is the record size in bytes.

Bounds (16 bytes): 0x00000000, 0x00000000, 0x00000059, 0x00000059 specify the rectangular inclusive-inclusive bounds of the smallest rectangle that can be drawn around the image stored in the metafile in logical units.

Frame (16 bytes): 0x00000000, 0x00000000, 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Signature (0x464D4520)																															
Version (0x00010000)																															
Byte (0x000037FC)																															
Records (0x00000016)																															
Handles (0x0005)																				Reserved (0x0000)											
nDescription (0x00000034)																															
offDescription (0x0000006C)																															
PalEntries (0x00000000)																															

Signature (4 bytes): 0x464D4520 is the record signature, which consists of the ASCII string "EMF".

Version (4 bytes): 0x00010000 specifies EMF metafile interoperability.

Bytes (4 bytes): 0x000037FC specifies the size of the metafile in bytes.

Records (4 bytes): 0x00000016 specifies the number of records in the metafile.

Handles (2 bytes): 0x0005 specifies the number of graphics objects that are 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 (2 bytes): 0x0000 is ignored.

nDescription (4 bytes): 0x00000034 specifies the number of characters in the array that contains the description of the metafile's contents.

offDescription (4 bytes): 0x0000006C specifies the offset from the beginning of this record to the array that contains the description of the metafile's contents.

PalEntries (4 bytes): 0x00000000 specifies the number of entries in the metafile palette. The location of the palette is specified in the EMR_EOF record.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Device (0x00000780)																															
... (0x00000780)																															
Millimeters (0x000002A5)																															
... (0x000001A7)																															
cbPixelFormat (0x00000000)																															
offPixelFormat (0x00000000)																															
bOpenGL (0x00000000)																															
MicrometersX (0x000A55D5)																															
MicrometersY (0x000675A5)																															
EmfDescription ("Sample EMF that has a brush fill, bitmap, and text")																															

Device (8 bytes): 0x00000780, 0x00000780 specify the size of the reference device, in pixels.

Millimeters (8 bytes): 0x000002A5, 0x000001A7 specify the size of the reference device, in millimeters.

cbPixelFormat (4 bytes): 0x00000000 specifies the size of the PixelFormatDescriptor (section 2.2.22) structure. This value indicates that no pixel format is defined.

offPixelFormat (4 bytes): 0x00000000 specifies the offset to the PixelFormatDescriptor in the metafile. In this case, no pixel format structure is present.

bOpenGL (4 bytes): 0x00000000 specifies that no OpenGL commands are present in the metafile.

Micrometers (8 bytes): 0x000A55D5, 0x000675A5 specify the horizontal and vertical size of the reference device, in micrometers.

EmfDescription (4 bytes): "Sample EMF that has a brush fill, bitmap, and text".

3.2.2 EMR_CREATEBRUSHINDIRECT Example

This section provides an example of an EMR_CREATEBRUSHINDIRECT record (section 2.3.7.1).

```
000000D0:            27 00 00 00 18 00 00 00 01 00 00 00
000000E0:02 00 00 00 52 47 2A 00 03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000027)																															
Size (0x00000018)																															
ihBrush (0x00000001)																															
LogBrush (0x00000002) (12 bytes)																															
... (0x0052472A)																															
... (0x00000003)																															

Type (4 bytes): 0x00000027 identifies this record type as EMR_CREATEBRUSHINDIRECT.

Size (4 bytes): 0x00000018 specifies the size of this record in bytes.

ihBrush (4 bytes): 0x00000001 specifies the index of this brush object in the EMF object table (section 3.1.1.1).

LogBrush (12 bytes): A LogBrushEx object (section 2.2.12) that contains brush data.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BrushStyle (0x00000002)																															
Color (0x0052472A)																															
BrushHatch (0x00000003)																															

BrushStyle (4 bytes): 0x00000002 specifies a hatch brush style, from the **BrushStyle** enumeration ([MS-WMF] section 2.1.1.4).

Color (4 bytes): 0x0052472A is a **ColorRef** object ([MS-WMF] section 2.2.2.8) that specifies the brush color value.

BrushHatch (4 bytes): 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 1

This section provides an example of an 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000001)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000001 specifies the index of an object in the EMF object table.

3.2.4 EMR_BITBLT Example 1

This section provides an example of an 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 80 3F
00000130:00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00
00000140:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150:00 00 00 00 00 00 00 00 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x0000004C)																															
Size (0x00000064)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0x00000059)																															
... (0x00000059)																															
xDest (0x00000000)																															
yDest (0x00000000)																															
cxDest (0x00000059)																															
cyDest (0x00000059)																															

Type (4 bytes): 0x0000004C identifies this record type as EMR_BITBLT.

Size (4 bytes): 0x00000064 specifies the size of this record in bytes.

Bounds (16 bytes): 0x00000000, 0x00000000, 0x00000059, 0x00000059 specify the bounding rectangle in logical units.

xDest (4 bytes): 0x00000000 specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): 0x00000000 specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): 0x0000005A specifies the logical width of the destination rectangle.

cyDest (4 bytes): 0x0000005A specifies the logical height of the destination rectangle.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BitBlitRasterOperation (0x00F00021)																															
xSrc (0x00000000)																															
ySrc (0x00000000)																															
xformSrc (0x3F800000) (24 bytes)																															
... (0x00000000)																															
... (0x00000000)																															
... (0x3F800000)																															
... (0x00000000)																															
... (0x00000059)																															

BitBlitRasterOperation (4 bytes): 0x00F00021 specifies the raster operation code from the **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 (4 bytes): 0x00000000 specifies the logical x-coordinate of the upper-left corner of the source rectangle.

ySrc (4 bytes): 0x00000000 specifies the logical y-coordinate of the upper-left corner of the source rectangle.

xformSrc (24 bytes): 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BkColorSrc (0x00000000)																															

UsageSrc (0x00000000)
offBmiSrc (0x00000000)
cbBmiSrc (0x00000000)
offBitsSrc (0x00000000)
cbBitsSrc (0x00000000)

BkColorSrc (4 bytes): 0x00000000 specifies the background RGB color.

UsageSrc (4 bytes): 0x00000000 specifies the value of the **Colors** field of the **DeviceIndependentBitmap** object ([MS-WMF] section 2.2.2.9) from the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): 0x00000000 specifies the offset to the source **DeviceIndependentBitmap** object.

cbBmiSrc (4 bytes): 0x00000000 specifies the size of the source **DeviceIndependentBitmap** object.

offBitsSrc (4 bytes): 0x00000000 specifies the offset to the source bitmap bits.

cbBitsSrc (4 bytes): 0x00000000 specifies the size of the source bitmap bits.

3.2.5 EMR_SELECTOBJECT Example 2

This section provides an example of an 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x80000000 = WHITE BRUSH)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x80000000 specifies the index of an object in the EMF object table.

3.2.6 EMR_BITBLT Example 2

This section provides an example of an 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 80 3F
000001A0:00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00
000001B0:00 00 00 00 FF FF FF 00 00 00 00 00 64 00 00 00
000001C0:28 00 00 00 8C 00 00 00 1C 2F 00 00 28 00 00 00
000001D0:59 00 00 00 2D 00 00 00 01 00 18 00 00 00 00 00
000001E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001F0:00 00 00 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
00000200:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F
00000210:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000220:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000230:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 3D
00000240:59 0C 5A 60 4E AE AE AE BF BF BF C5 C5 C5 C0 C0
00000250:C1 B1 B2 B1 A1 A1 A1 A9 AA AA AE AF B0 A2 A2 A2
00000260:A6 A6 A4 AF AE AD AC AC AC A6 A6 A6 99 99 99 7D
00000270:7D 7D 66 66 65 5A 5A 59 4F 4F 4F 58 58 58 76 76
00000280:76 9E 9E 9E B5 B5 B5 3C 43 2D 32 46 0A 34 4A 0A
00000290:34 49 0A 36 4C 0B 3A 52 0B 3F 59 0C 41 5C 0D 42
000002A0:5D 0D 42 5D 0D 44 5D 0D 44 5D 0D 43 5C 0C 43 5C
000002B0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C
000002C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000002D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000002E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000002F0:45 5B 0C 45 5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00
00000300:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00000310:5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E
00000320:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000330:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41
00000340:5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 39 52 0C 70 73
00000350:6B AD AD AD A8 A8 A8 99 99 98 A6 A6 A6 A6 A7 A7
00000360:A3 A5 A6 A6 A5 A3 AE A5 9D C9 BB AD A1 9B 95 A2
00000370:9F 9C B3 B1 AF B6 B4 B3 A9 A9 A9 97 93 90 8E 87
00000380:81 89 84 7F 81 7E 7A 79 78 76 80 7F 7E 98 98 98
00000390:BC BC BC 6D 70 65 33 47 0A 37 4D 0B 35 4B 0A 35
000003A0:4B 0B 38 4F 0B 3D 56 0C 41 5B 0D 42 5D 0D 42 5D
000003B0:0D 43 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 43 5C 0C
000003C0:42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000003D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000003E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000003F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45
00000400:5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F
00000410:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F
00000420:0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E
00000430:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41
00000440:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000450:0E 41 5E 0E 40 5D 0D 38 51 0E 90 91 8E AC AC AC
00000460:C3 C3 C3 C5 C6 C6 B9 BB BC B3 B0 AD BA AD 9E C8
00000470:B4 A0 DB C8 B4 E4 D1 BE C2 B5 A9 82 7E 7A 8C 88
00000480:84 83 80 7E 98 8F 85 BE AC 9A C9 B6 A3 CC BB A9
00000490:CD C0 B3 AB A3 9B 80 7E 76 78 76 70 91 8F 8D 84
000004A0:86 80 36 4B 0B 39 50 0B 38 4E 0B 37 4D 0B 38 4F
000004B0:0B 3C 55 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D 0D
000004C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43
000004D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000004E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000004F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000500:5C 0C 43 5C 0C 43 5C 0C 45 5B 0A 45 5B 0A 45 5C
00000510:0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F
00000520:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D
00000530:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41
00000540:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000550:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000560:40 5D 0D 3D 50 1C 94 94 94 A1 A1 A0 C9 CA CA BC
00000570:BB B8 BB AC 9C C4 AD 94 D2 BB A4 E0 CF BD E0 D0
00000580:BF DA C7 B5 DE CB B8 9A 93 8B 7C 76 70 A2 93 83
00000590:C0 AB 95 C6 B2 9D CA B8 A4 D1 C0 AD E0 D2 C4 E8
000005A0:DC CF CD C3 B5 8A 85 74 72 72 61 83 82 7E 34 43
```

```
000005B0:15 3C 54 0C 38 50 0B 38 4E 0B 39 50 0B 3D 55 0C
000005C0:40 5B 0D 42 5D 0D 42 5D 0D 42 5D 0D 43 5C 0C 43
000005D0:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C
000005E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000005F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000600:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000610:0C 43 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A
00000620:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
00000630:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F
00000640:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000650:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000660:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 4B
00000670:56 33 B8 B8 B8 B7 B7 B7 A1 A1 A2 BC AC 9B CC B2
00000680:97 D6 C3 AE E5 D6 C6 E4 D4 C4 E1 D2 C2 D9 C6 B4
00000690:D7 C3 AD CE BF AF AD 9A 85 BB A5 8D BD A9 93 C3
000006A0:AF 9B C8 B5 A2 CE BD AA DC CE C0 E2 D5 C9 E1 D4
000006B0:C7 D5 C5 B4 B1 A9 95 91 97 7B 4A 57 2D 39 51 0B
000006C0:38 50 0B 38 50 0B 3A 52 0C 3E 58 0C 41 5C 0D 42
000006D0:5D 0D 42 5D 0D 42 5D 0D 41 5C 0C 43 5C 0C 43 5C
000006E0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C
000006F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000700:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000710:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A
00000720:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 00
00000730:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00000740:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E
00000750:0F 41 5E 0F 41 5E 0F 41 5E 0E 41 5E 0E 40 5D 0D
00000760:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40
00000770:5D 0D 40 5D 0D 41 5E 0E 41 5E 0E 61 6C 44 D0 D0
00000780:D0 B0 B0 B0 C8 C3 BE DB C8 B3 DC CA B8 EB DD CF
00000790:E8 D9 CA E5 D6 C7 E4 D6 C8 D9 C8 B5 D2 BF AA DD
000007A0:CC BA C7 B3 9E B6 9F 89 BC A7 92 C1 AD 98 C6 B3
000007B0:9F CC BA A8 DB CD BF DD D0 C2 DF D3 C6 E0 D3 C5
000007C0:E2 D4 C4 A7 AA 8E 52 62 2B 37 4D 0B 39 51 0B 3B
000007D0:53 0C 3E 57 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D
000007E0:0D 42 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 45 5B 0D
000007F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000800:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000810:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000820:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45
00000830:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F
00000840:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F
00000850:0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0F
00000860:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00000870:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000880:0D 41 5E 0E 3F 5B 0D 6A 79 47 D1 D2 CE C5 C2 BF
00000890:EF E6 DD E7 D8 CA EE E0 D3 ED DF D2 EA DC CE E9
000008A0:DB CD E1 D1 C1 D6 C3 B0 D7 C4 B1 DA C9 B8 D5 C3
000008B0:B1 B8 A3 8C BC A7 91 C0 AC 97 C5 B1 9D CB B9 A5
000008C0:D6 C8 B8 CE BF AE D9 CB BD E1 D4 C7 E3 D4 C6 BC
000008D0:B9 A2 53 63 2C 37 4C 0B 3C 55 0C 3E 58 0C 40 5B
000008E0:0D 41 5C 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D
000008F0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43
00000900:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000910:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000920:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000930:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C
00000940:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F
00000950:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D
00000960:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40
00000970:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E
00000980:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E
00000990:40 59 0F 75 84 54 A6 B1 8E DC D9 CE F5 EE E8 E7
000009A0:DA CE E5 D7 C8 EB DD D0 EC DF D2 E4 D5 C4 D8 C8
000009B0:B5 D9 CA B8 D9 CB BA D9 C8 B6 D9 C8 B6 C4 B0 9C
000009C0:BA A5 90 BF AA 95 C3 AF 9B C9 B6 A3 C9 B9 A7 C7
000009D0:B7 A6 D1 C2 B3 DC CF C0 E0 D2 C3 CE C4 B3 5B 69
000009E0:33 3C 52 0D 40 5A 0D 41 5C 0D 42 5D 0D 42 5D 0D
000009F0:42 5D 0D 42 5D 0D 41 5C 0C 42 5D 0D 43 5C 0C 43
00000A00:5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C
```

```
00000A10:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000A20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000A30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000A40:0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A
00000A50:45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
00000A60:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F
00000A70:5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D
00000A80:0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D
00000A90:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 44 5C 12 8B
00000AA0:97 6A 93 A2 76 C8 CA B5 F5 EE E9 E8 DC CF E0 D0
00000AB0:C1 DF D2 C4 E1 BB A1 D4 AC 8E D5 B2 97 D3 A4 84
00000AC0:D3 A5 86 D7 C4 B0 D7 C5 B1 CF BB A6 C2 AE 99 C1
00000AD0:AD 98 C1 AE 99 C5 B0 9B C7 B5 A2 C8 B8 A7 D1 C2
00000AE0:B2 E0 D3 C6 E4 D8 CA E2 D5 C6 71 7B 4B 42 58 10
00000AF0:42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42
00000B00:5D 0D 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000B10:0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000B20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000B30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000B40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A
00000B50:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00
00000B60:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00000B70:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E
00000B80:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00000B90:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40
00000BA0:5D 0D 40 5D 0D 41 5E 0E 4B 60 17 A0 AB 83 92 A2
00000BB0:73 AC B6 94 F4 ED E7 E2 CF BE DA CA BA D7 C4 B2
00000BC0:D0 86 58 CA 71 3D C9 69 32 C6 5D 24 C4 54 17 CC
00000BD0:86 5D D5 C5 B2 D2 BC A6 C9 B3 9D C2 AD 96 C1 AC
00000BE0:97 C8 B1 9A D9 C8 B6 D3 C4 B4 D1 C3 B3 DE D1 C4
00000BF0:E0 D3 C5 E3 D5 C6 A3 A4 81 48 5E 13 42 5D 0D 42
00000C00:5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 41 5C
00000C10:0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C
00000C20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000C30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000C40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000C50:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45
00000C60:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F
00000C70:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F
00000C80:0E 40 5D 0D 3E 5E 0D 3E 5E 0D 40 5D 0D 40 5D 0D
00000C90:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00000CA0:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000CB0:0D 41 5E 0E 52 64 1F A4 AF 8A 95 A3 75 B3 AC 84
00000CC0:DC AB 87 D7 A3 7D DB CD BD D2 A1 7F CD 76 42 CB
00000CD0:71 3C C9 69 32 C7 62 2A C5 5B 20 C3 54 19 D8 AD
00000CE0:90 E2 D7 CA D3 BE A9 CA B6 A0 D0 BD AA D1 BD A9
00000CF0:DD CD BD DD D0 C2 D0 C1 B1 DA CD BE DE D1 C3 DB
00000D00:CD BC 91 9A 6E 4E 62 18 42 5D 0D 42 5D 0D 42 5D
00000D10:0D 42 5D 0D 42 5E 0B 41 5C 0C 41 5C 0C 41 5C 0C
00000D20:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43
00000D30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000D40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000D50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000D60:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C
00000D70:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F
00000D80:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 3E 5D 0D
00000D90:3E 5D 0D 3E 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00000DA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000DB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0E
00000DC0:58 67 26 A4 B1 8B B8 B1 8A D6 9F 74 D5 98 6B D7
00000DD0:A4 7F DD CF BF D2 90 66 CD 77 43 CA 6E 38 C8 65
00000DE0:2C C6 60 27 C6 5B 21 C3 56 1B CB 71 40 DD C8 B6
00000DF0:D5 C3 AF E1 D2 C2 DF D0 C1 D9 C8 B7 E1 D3 C4 E4
00000E00:D8 CA D1 C2 B2 E0 D4 C7 E7 DA CD AD AA 90 6F 7F
00000E10:47 52 63 1F 41 5B 0D 41 5C 0C 41 5C 0C 42 5D 0D
00000E20:42 5D 0B 41 5C 0C 41 5C 0C 41 5B 0C 42 5B 0C 43
00000E30:5B 0C 43 5C 0C 43 5B 0B 43 5C 0C 43 5C 0C 43 5B
00000E40:0C 43 5B 0C 43 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000E50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000E60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
```

```
00000E70:0C 44 5C 0A 44 5C 0A 45 5B 0A 45 5B 0A 45 5B 0A
00000E80:45 5B 0B 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
00000E90:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40
00000EA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000EB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E
00000EC0:40 5D 0D 40 5D 0D 40 5D 0D 42 5C 0E 60 6E 33 A9
00000ED0:B6 91 CD B0 8B D7 A2 78 D6 99 6D D9 A8 86 E2 D7
00000EE0:CA DE BC A2 CC 78 44 D0 85 57 D2 8B 61 CC 74 42
00000EF0:CC 76 46 CD 7D 4F CC 76 47 CF 9C 7A D8 C9 B7 DF
00000F00:D0 C0 DF CF BF DD CE BE E4 D7 C9 EB E0 D4 D1 C3
00000F10:B3 DC CF C1 F1 E6 DA C8 C0 AE 72 80 4F 56 66 24
00000F20:42 5B 0E 41 5C 0C 41 5C 0C 42 5D 0D 43 5C 0C 43
00000F30:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C
00000F40:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D
00000F50:43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00000F60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00000F70:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00000F80:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00
00000F90:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00000FA0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00000FB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00000FC0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40
00000FD0:5D 0D 40 5D 0D 43 5C 10 78 82 4C B9 BF 9C D8 AB
00000FE0:84 D6 A3 7A D6 A2 7C DE BE A5 E7 DA CD E8 DA CB
00000FF0:DA AA 89 E7 D9 CC DB AE 91 C7 63 2B C6 5D 24 C4
00001000:56 1B C2 51 15 D1 A2 83 E1 D4 C5 E1 D2 C2 E1 D1
00001010:C2 D1 C0 AD D1 C2 B1 E2 D6 C9 D3 C5 B5 D4 C6 B7
00001020:E2 D6 C8 EA DE D2 95 9A 76 5B 6B 2A 44 5C 0F 41
00001030:5C 0C 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001040:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B
00001050:43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43
00001060:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001070:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00001080:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45
00001090:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F
000010A0:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F
000010B0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000010C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000010D0:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D
000010E0:0D 47 5E 13 8E 97 67 C4 BF 9F CE A9 81 CA B7 99
000010F0:E4 DA CF DB CC BC E9 DA CC E8 DA CC E9 DB CE E5
00001100:CF BE CB 73 40 C6 5F 26 C6 5C 22 C3 52 15 CA 75
00001110:45 D9 CA B9 E6 D8 CA E3 D5 C6 D1 BF AE C3 AF 9A
00001120:CE BE AD D1 C3 B3 CC BD AD D2 C4 B5 D9 CC BD DD
00001130:CE BF CE C6 B3 6A 78 3A 47 5C 11 41 5C 0C 41 5C
00001140:0C 42 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D
00001150:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43
00001160:5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43 5C 0C 43 5C
00001170:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00001180:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001190:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B
000011A0:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F
000011B0:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D
000011C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000011D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000011E0:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 4C 61 17
000011F0:9C A5 7A D1 BF 9E D6 A5 7D E2 BD 9F E3 CF BE DB
00001200:CB BB E9 DA CC E8 D9 CB EA DF D3 DC B0 93 C7 63
00001210:29 C5 5C 22 C5 59 1F C5 5B 21 DD BC A4 DE D2 C4
00001220:E6 D8 CA E8 DA CC CD BB A9 C6 B3 9F CE BD AB CA
00001230:B8 A5 CF BE AC D5 C4 B2 BD B4 9D AB AC 8B B6 BA
00001240:9A 78 86 4D 4A 5E 14 43 5C 0C 43 5C 0C 43 5C 0C
00001250:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43
00001260:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B
00001270:0D 42 5A 0C 43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C
00001280:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001290:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000012A0:0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C
000012B0:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
000012C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40
```

```
000012D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000012E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E
000012F0:40 5D 0D 40 5D 0D 40 5D 0D 53 64 1D A2 AA 82 D7
00001300:BC 99 D8 A7 7F DA A8 81 E3 CE BB E1 D2 C3 E8 D9
00001310:CC E8 D9 CB E4 D6 C7 E6 D7 C9 E2 BC A3 D2 84 57
00001320:C5 56 1A D4 9A 76 E6 DE D2 E0 D0 C1 E3 D5 C6 F0
00001330:E4 D7 E1 D2 C3 CA B7 A4 D1 C1 B0 C9 B7 A3 D0 BD
00001340:A9 D8 C6 B2 B5 AC 92 7B 89 53 75 89 4D 70 82 46
00001350:53 61 1C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001360:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C
00001370:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C
00001380:42 5A 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001390:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000013A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000013B0:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00
000013C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
000013D0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000013E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000013F0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40
00001400:5D 0D 41 5D 0E 59 67 24 A9 B2 8C D1 BB 98 D7 A5
00001410:7C D4 B0 8D DE D4 C5 DA CA BB DE D1 C5 DB D0 C3
00001420:DC CE BF E4 D5 C5 EC E2 D7 F0 E5 D9 E4 C4 AE D7
00001430:C8 B9 D4 C7 BA D8 C9 BB E0 D2 C4 E9 DD D0 E3 D4
00001440:C6 CF BD AB D2 C3 B3 C2 B2 A1 C6 B6 A3 CD BA A7
00001450:CD BD A7 8B 97 65 7D 8E 55 74 85 49 5A 66 21 44
00001460:5C 0E 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001470:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B
00001480:43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 42 5A 0C 42
00001490:5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000014A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000014B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45
000014C0:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F
000014D0:5F 0E 3F 5F 0C 3F 5F 0C 3F 5F 0C 3F 5F 0E 40 5E
000014E0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000014F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001500:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 43 5D
00001510:0F 69 71 33 B3 BC 99 CD BA 96 D6 A4 79 A2 9B 72
00001520:B3 AE 91 DE D0 C2 D9 C2 AF CE 9D 7C CF 8E 64 CE
00001530:7D 4C D6 AB 8D DD D4 C9 DB D1 C6 DB CF C2 D8 CA
00001540:BC CF C1 B2 D1 C4 B6 DB CF C3 D6 C8 BB C6 B6 A5
00001550:CF C2 B4 D4 C8 BC A1 9B 81 69 6C 3C 60 63 2E 6C
00001560:77 41 7A 88 54 74 83 4B 60 6C 27 45 5C 0E 43 5C
00001570:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D
00001580:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43
00001590:5C 0C 43 5B 0D 44 5A 0C 44 5A 0C 44 5B 0B 43 5C
000015A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000015B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000015C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0C 43 5B
000015D0:0C 45 5B 0C 43 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60
000015E0:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D
000015F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001600:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001610:0D 40 5D 0D 40 5D 0D 40 5D 0D 45 5E 10 7D 82 48
00001620:BE C5 A4 B9 B0 8A BA B2 93 D1 C9 B6 DB AC 89 E6
00001630:CC B7 E8 D2 C0 D1 85 55 CC 75 41 CA 6C 36 C6 6E
00001640:3B B5 A4 8E D2 C6 B8 E7 D8 C9 E5 D6 C7 E0 D1 C2
00001650:DA CC BD DB CD C0 D3 C6 B7 BE AF A0 D0 C3 B5 E9
00001660:DE D2 EA E2 D6 C3 BE A9 8B 89 64 61 66 34 65 6D
00001670:3E 68 76 45 63 6C 2D 47 5D 0F 43 5C 0C 43 5C 0C
00001680:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43
00001690:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B
000016A0:0D 45 5B 0D 45 5B 0C 45 5B 0C 43 5C 0C 42 5B 0B
000016B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000016C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000016D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C
000016E0:43 5D 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C
000016F0:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40
00001700:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001710:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001720:40 5D 0D 40 5D 0D 47 5E 12 91 93 61 BC C2 A1 C3
```

```
00001730:C4 AB E8 E0 D4 DF B4 93 D4 96 68 DE B2 93 E0 BB
00001740:A0 CF 81 50 CD 7A 48 CB 71 3C C8 63 2A D3 9A 75
00001750:E1 D5 C7 DB CB BB D8 C7 B7 D7 C6 B5 D8 C7 B6 D8
00001760:C7 B6 D3 C1 AF C5 B2 9E DB CD BE E9 DE D3 EF E5
00001770:DB F6 EE E4 FA F0 E7 E3 D8 CA BB B0 9A A1 9D 83
00001780:71 74 3B 4A 5E 12 43 5C 0C 43 5C 0C 43 5C 0C 43
00001790:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C
000017A0:0C 42 5B 0B 43 5B 0C 43 5B 0C 43 5C 0C 45 5B 0C
000017B0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43
000017C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000017D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000017E0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0A 00
000017F0:3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F
00001800:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001810:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001820:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001830:5D 0D 4E 60 17 9B 9C 6F BB C2 9E D7 D5 C2 E0 C0
00001840:A6 D7 A0 76 D6 9C 72 D8 A1 7A D4 95 6A D0 82 51
00001850:CD 7A 47 CA 6D 38 D7 A7 88 DB BF AA D9 C9 B8 D3
00001860:C3 B2 D0 BF AC D0 BE AC D1 C0 AE CF BD AA C6 B3
00001870:9F D1 BF AD E2 D5 C7 E6 DC D0 EA DF D4 E9 DE D3
00001880:E4 D8 CC DC CE C0 D1 C0 AF CC BD AA 73 79 3E 50
00001890:5F 17 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000018A0:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B
000018B0:43 5B 0D 43 5B 0D 43 5C 0C 45 5B 0C 44 5A 0B 44
000018C0:5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C
000018D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000018E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000018F0:5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F
00001900:5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E
00001910:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001920:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001930:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 57 64
00001940:1E A5 A5 7C B8 BE 9B D4 D1 BF E2 C5 AD D8 A2 78
00001950:D7 9E 74 D8 A4 7E D2 8C 5E D0 83 53 CC 74 40 D1
00001960:8D 62 DE D0 C2 DC CE C0 D4 C2 B1 CF BD AB D1 BF
00001970:AD D4 C3 B2 D4 C3 B1 CF BE AB D1 BF AD DC CC BC
00001980:DE D0 C3 DF D3 C6 DD D0 C3 DC CF C1 DF D3 C5 DA
00001990:CC BF D3 C5 B6 90 8D 58 60 6D 27 56 64 1E 43 5C
000019A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D
000019B0:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 43
000019C0:5B 0D 43 5C 0C 44 5A 0B 44 5A 0B 44 5A 0B 42 5B
000019D0:0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000019E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000019F0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001A00:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60
00001A10:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D
00001A20:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001A30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001A40:0D 40 5D 0D 40 5D 0D 40 5D 0D 62 6B 26 B0 B1 8C
00001A50:BB BE A0 E3 DD D0 E8 D7 C8 D8 A2 7A D7 A4 7D D5
00001A60:9C 74 D1 87 57 CE 7C 4A D0 88 5C D7 C2 AF D4 C5
00001A70:B5 D5 C4 B2 D9 C9 B8 DD CD BD DC CC BC DA C9 B9
00001A80:D7 C6 B6 D7 C6 B5 DF D0 C0 D7 C7 B6 D7 C8 B9 D5
00001A90:C7 B9 D9 CB BD E5 DA CD E4 D9 CC E0 D3 C6 D7 CD
00001AA0:BD 7C 84 48 5B 6A 1F 5C 67 21 43 5C 0C 43 5C 0C
00001AB0:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43
00001AC0:5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 42 5A 0C 42 5B
00001AD0:0B 44 5A 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B
00001AE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001AF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001B00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C
00001B10:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C
00001B20:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40
00001B30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001B40:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001B50:40 5D 0D 41 5D 0D 6D 73 32 B9 BA 9B D6 D1 BF EC
00001B60:E3 DA EE E7 DF E1 C0 A5 D7 AB 88 D7 A3 7E DA A7
00001B70:84 DD B1 94 D9 C4 B1 D4 C5 B4 DB CB BB E9 DB CD
00001B80:E9 DB CE E3 D5 C6 DE CF BF DC CD BC DF CF 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 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001C00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00001C10:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00
00001C20:40 5D 0D 41 5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00001C30:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001C40:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001C50:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41
00001C60:5E 0F 77 79 3C CC CA B4 EB E2 D9 ED E4 DB EF E7
00001C70:E0 EF E7 DE DE CC BB DD CD BC E8 DB CD E9 DC CF
00001C80:DC CD BD E5 D6 C7 F1 E4 D8 F0 E3 D6 EA DC CE E5
00001C90:D6 C7 E1 D2 C2 DF CF C0 DF CF C0 DD CD BD D9 C8
00001CA0:B8 D7 C6 B5 E0 D3 C6 E6 DB CF EB E0 D5 E1 D5 C8
00001CB0:DF D3 C7 E4 D8 CB A7 A3 88 77 82 4F 77 86 4B 6B
00001CC0:71 30 44 5D 0E 41 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001CD0:0C 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D
00001CE0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44
00001CF0:5A 0B 42 5B 0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C
00001D00:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C
00001D10:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
00001D20:5C 0C 43 5C 0C 42 5B 0B 44 5B 09 00 40 5D 0D 41
00001D30:5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E
00001D40:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E
00001D50:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00001D60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 47 5F 13 80 7F
00001D70:46 DE DA C9 EF E6 DD EE E5 DD F0 E8 E0 F1 E8 E0
00001D80:E2 D2 C3 DE CD BD E3 D3 C3 E3 D4 C5 E2 D3 C5 E6
00001D90:D9 CA E5 D7 C9 E7 DA CB E9 DA CC E5 D6 C8 E2 D3
00001DA0:C4 E0 D1 C2 DA CA B9 D3 C2 B1 CF BD AA CC B9 A6
00001DB0:D6 C7 B8 E1 D5 C9 E7 DB CF DF D3 C6 E0 D3 C7 DE
00001DC0:D1 C5 D4 C9 B9 7F 85 5B 75 83 4A 74 78 3A 4A 5F
00001DD0:14 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00001DE0:42 5B 0B 42 5B 0C 42 5B 0C 43 5B 0C 42 5B 0B 42
00001DF0:5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B 42 5B
00001E00:0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A
00001E10:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43
00001E20:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001E30:0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 5E 0E 3F 5F
00001E40:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E 0E 40 5D 0D
00001E50:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40
00001E60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00001E70:0D 40 5D 0D 40 5D 0D 4B 61 15 8E 8B 57 CF CE B6
00001E80:EA E2 D6 F0 E7 DF F1 E9 E1 F1 E8 E0 E4 D5 C5 E1
00001E90:D1 C0 E0 D0 C0 E0 D0 C2 E4 D5 C6 E3 D4 C5 E3 D4
00001EA0:C5 E5 D6 C8 E8 DB CD EB DD D0 EB DD CF E8 DA CC
00001EB0:E0 D1 C2 D7 C6 B5 CE BB A9 C9 B5 A2 D3 C4 B5 D3
00001EC0:C4 B6 DA CD BF E1 D5 C9 E3 D7 CB DD D1 C4 EA DF
00001ED0:D4 B7 B4 9C 77 85 4D 76 7D 3E 4F 61 17 41 5C 0C
00001EE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 42
00001EF0:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00001F00:0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B
00001F10:44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43
00001F20:5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00001F30:0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5A 0B
00001F40:44 5B 09 00 40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E
00001F50:3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40
00001F60:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D
00001F70:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00001F80:40 5D 0D 50 63 1A A3 9F 71 CA C9 B1 E0 D9 CB F1
00001F90:E8 E0 F2 EA E2 F2 E9 E0 E7 D8 C8 E4 D4 C4 E1 D0
00001FA0:C1 E4 D5 C6 DD CD BD D7 C6 B6 DE CF BF E5 D6 C8
00001FB0:EB DE D0 F0 E3 D6 F1 E5 D8 F0 E3 D6 ED DF D3 E9
00001FC0:DB CD E8 DA CC E3 D4 C5 E3 D7 CA E4 D8 CC D9 CB
00001FD0:BD D9 CB BD E0 D3 C7 DF D3 C6 E7 DC D0 E1 D8 C8
00001FE0:80 8D 56 7A 82 45 55 64 1B 41 5C 0C 43 5C 0C 43
```

```
00001FF0:5C 0C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002000:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002010:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43
00002020:5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C
00002030:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00002040:43 5C 0C 43 5C 0C 43 5A 0B 44 5A 0B 44 5B 09 00
00002050:40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00002060:5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
00002070:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00002080:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 58
00002090:67 20 B2 AF 88 D8 D4 C5 F0 E7 DE F1 E9 E1 F3 EC
000020A0:E4 F3 E9 E1 E9 DA CB E6 D6 C7 E4 D4 C4 E6 D8 C9
000020B0:DF D0 C0 D5 C4 B3 DB CB BA E5 D7 C9 EB DD D0 EF
000020C0:E2 D5 F1 E4 D8 F0 E4 D7 ED DF D2 EE E1 D4 E6 D8
000020D0:C9 C1 AE 98 CA B9 A8 E8 DC D1 EC E1 D7 E7 DC D0
000020E0:DD D0 C3 D4 C6 B7 D8 CB BC AE AC 8B 79 87 4D 7B
000020F0:84 4A 5C 67 20 41 5C 0C 43 5C 0C 43 5C 0C 43 5C
00002100:0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002110:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002120:5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C
00002130:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C
00002140:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42
00002150:5B 0B 44 5A 0B 44 5A 0B 44 5B 09 00 40 5D 0D 40
00002160:5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D
00002170:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E
00002180:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00002190:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 64 6C 29 C3 BD
000021A0:A0 F4 EC E3 F2 E9 E0 F3 EA E2 F5 EE E6 F3 EA E1
000021B0:EB DC CD E8 D9 CA E5 D5 C6 E4 D4 C5 E3 D3 C3 D9
000021C0:C8 B8 D3 C1 AF DC CC BC EB DD D0 EF E2 D6 F2 E5
000021D0:DA F2 E5 D9 F4 E7 DB EA DC CE C7 B3 A0 B7 A2 8B
000021E0:D1 C2 B3 EB E0 D6 E6 DA CF EA E0 D5 EF E4 DB E8
000021F0:DC D1 D0 C5 B4 8A 8F 63 7C 8B 51 79 85 4A 64 69
00002200:27 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B
00002210:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002220:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A
00002230:0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A
00002240:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43
00002250:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 44 5A 0B 44 5A
00002260:0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 5D 0D 3F 5E
00002270:0D 3F 5E 0D 3F 5E 0D 3F 5E 0D 40 5D 0D 40 5D 0D
00002280:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00002290:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000022A0:0D 3F 5C 0C 40 5C 0D 72 74 35 CC C5 AC F4 EC E4
000022B0:F1 E9 E0 F6 EE E6 F5 EE E6 F2 E9 DE ED DE CF EB
000022C0:DC CE E3 D3 C3 D8 C7 B4 DB CB BA DF D0 BF D9 C8
000022D0:B7 D9 C8 B7 ED DE D0 F4 E6 D9 F9 ED E0 FD F0 E4
000022E0:FF F3 E7 E0 CF BD CA B6 A1 C7 B3 9D F3 E7 DC F4
000022F0:EA E0 E7 DD D0 E7 DC D0 E8 DD D2 EF E5 DB F0 E7
00002300:DD A5 A6 81 7C 8A 50 76 83 47 6F 71 31 43 5C 0E
00002310:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002320:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002330:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B
00002340:43 5A 0B 42 5B 0B 42 5B 0B 43 5C 0A 43 5C 0A 43
00002350:5C 0A 43 5C 0C 43 5C 0C 43 5C 0B 43 5C 0B 43 5C
00002360:0C 43 5B 0C 42 5A 0A 43 59 0A 44 5A 0B 44 5A 0B
00002370:43 5B 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
00002380:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00002390:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000023A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C
000023B0:43 5E 0F 80 7E 41 D3 CD BA F2 E9 E0 F4 EB E2 F4
000023C0:EA E1 F0 E8 DF E5 D9 CD DD CE BE E4 D5 C7 DC CD
000023D0:BF D9 CA BB D9 CA BB DA CC BD D2 C4 B6 CC BF B2
000023E0:CC C1 B5 C8 BD B2 C3 B9 AF C0 B6 AB BA AE A2 AA
000023F0:9A 8B A6 95 84 BB AD 9E D7 CE C5 D5 CA BF D3 C6
00002400:B9 DC CF C1 E5 D7 C9 EB DF D2 E6 DC CF 8D 92 67
00002410:7A 87 4D 73 82 45 79 79 39 49 5E 11 43 5C 0C 42
00002420:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002430:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002440:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42
```

```
00002450:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00002460:0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B
00002470:43 59 0A 43 59 0A 44 5A 0B 44 5A 0B 42 5C 09 00
00002480:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
00002490:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000024A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000024B0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 45 5F 11 90
000024C0:8A 4F EA E1 D5 E8 DF D7 E7 DF D6 E7 E1 DA E5 E0
000024D0:DB E6 E3 DF EA E8 E5 F0 EE ED F2 F2 F1 F1 F1 F1
000024E0:F0 F0 F0 EB EB EB E3 E4 E4 D7 D7 D8 C2 C3 C4 A8
000024F0:A9 AA 8B 8B 8C 6C 6C 6D 53 54 55 45 45 46 3E 3F
00002500:3F 3B 3B 3B 3C 3B 3A 41 40 3F 4A 48 46 5B 57 54
00002510:78 72 6C 9C 94 8B B4 AB A2 A2 9F 86 95 9C 6F 7A
00002520:8A 4F 7B 7D 3D 4B 5F 13 43 5C 0C 42 5B 0B 42 5B
00002530:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002540:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002550:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B
00002560:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
00002570:43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B 43 59 0A 43
00002580:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0D 40
00002590:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000025A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000025B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000025C0:5D 0D 40 5D 0D 3F 5C 0C 52 64 1D A1 92 80 C5 C5
000025D0:C4 D2 D2 D2 E1 E2 E3 EB EC ED F2 F2 F3 F5 F5 F5
000025E0:F7 F7 F6 F6 F6 F6 F5 F4 F3 F4 F3 F1 F1 EF EE F1
000025F0:EF ED ED EC EA E7 E5 E3 DF DE DC D2 D1 D0 C4 C2
00002600:C0 B7 B5 B2 AB A9 A6 A0 9D 9B 93 90 8D 89 88 87
00002610:7E 7E 7F 72 72 71 6B 6A 6A 69 69 6A 69 6A 6B 6F
00002620:71 73 83 86 89 A2 A3 A5 BB BA BA C4 C7 BF AE A4
00002630:81 55 64 1C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B
00002640:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002650:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A
00002660:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C
00002670:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43
00002680:5D 0A 43 5C 0C 44 5A 0B 44 5A 0B 43 59 0A 43 59
00002690:0A 44 5A 0B 42 5C 09 00 40 5D 0D 40 5D 0D 40 5D
000026A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000026B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000026C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000026D0:0D 3F 5C 0C 75 82 5E 9E A0 A2 C6 C7 C8 DE DE DE
000026E0:EA E9 E8 F2 F1 F0 F5 F3 F0 F5 F3 F0 F1 ED E9 EB
000026F0:E6 E0 E9 E3 DD E6 E1 DA E0 DB D6 E9 E5 DD F6 F4
00002700:F3 E9 E5 E1 E4 DF DA DC D7 D0 D2 CA C1 D4 CC C4
00002710:DB D4 CC E2 DC D5 D4 CC C3 CF C7 BC F3 EF EB DF
00002720:D9 D2 D9 D2 CB DC D7 D1 DB D8 D4 C9 C4 B8 B8 AF
00002730:96 C3 BF B7 D0 D0 D0 D0 D1 D2 CF D1 D3 86 8F 6D
00002740:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002750:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002760:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B
00002770:42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43
00002780:5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A
00002790:0B 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 44 5A 0B
000027A0:42 5C 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000027B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000027C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000027D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C
000027E0:89 91 7B AD AC AC AD A4 84 A2 97 5E A7 9B 5F EE
000027F0:E9 DA FC FC FC F8 F6 F4 EE E9 E4 E5 DE D7 E6 E1
00002800:DB E4 E0 DB B8 AA 7D B7 AD 7C FF FF FF F2 F2 F2
00002810:E5 E1 DD DE D9 D3 D6 CF C8 DB D6 D0 DB D6 D0 D9
00002820:D3 CD D1 CA C2 BE B4 A8 BA AF A3 C6 BD B3 CE C6
00002830:BE D1 CD C7 C5 B8 99 9A 8F 4D 91 88 43 9A 90 4E
00002840:A5 99 61 B2 A9 8B CB CB CC 9D A4 90 43 5C 0C 42
00002850:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002860:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002870:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42
00002880:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
00002890:0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A 0B 44 5A 0B
000028A0:43 59 0A 43 59 0A 43 59 0A 44 5A 0B 42 5C 09 00
```

```
000028B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000028C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D
000028D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000028E0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 90 99 7F 7D
000028F0:7C 7C 73 70 69 83 7D 64 93 88 5F A6 9C 78 EA E8
00002900:E4 FB F9 F7 E7 E1 DA E1 D9 D1 D9 D0 BF AB 9F 68
00002910:8D 84 3E 99 8F 4C C8 BB 90 E6 E0 CA F3 F0 EB E9
00002920:E5 E1 E0 DA D4 E6 E2 DE DD D7 D2 CE C7 BF DF DA
00002930:D5 E1 DC D7 D5 CE C6 D6 CF C7 CC C3 B4 BC B1 99
00002940:96 8A 54 8C 80 47 90 84 56 8E 83 5D 89 82 6C 9A
00002950:97 91 B2 B2 B3 92 9B 80 43 5C 0C 42 5B 0B 42 5B
00002960:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002970:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002980:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B
00002990:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000029A0:43 5D 0A 42 5C 09 44 5A 0B 44 5A 0B 43 59 0A 43
000029B0:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0B 40
000029C0:5D 0B 40 5E 0C 40 5E 0C 40 5E 0C 40 5E 0C 40 5E
000029D0:0C 40 5D 0B 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D
000029E0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40
000029F0:5D 0D 40 5D 0D 3F 5C 0C 49 63 1A 85 94 6B A3 A7
00002A00:9E 9F A1 A3 9E A1 A5 A3 A4 A6 B9 B8 B7 EA E7 E5
00002A10:DF D6 CD CE C8 C2 B6 B0 A3 A8 9F 81 A7 9F 81 A2
00002A20:98 77 9A 8F 66 9A 91 6B A2 9B 84 B5 B4 AB D4 D1
00002A30:CD DE D9 D3 DE D8 D1 D0 C9 C1 D0 CA C4 DF DD DC
00002A40:E4 E1 DE CB C8 C6 7E 7A 71 57 52 49 56 53 50 5B
00002A50:5C 5E 6E 71 74 86 88 8B A0 A1 A3 A6 AB A0 82 8F
00002A60:66 4C 63 18 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B
00002A70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002A80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A
00002A90:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C
00002AA0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 09 43
00002AB0:5B 09 44 5A 09 44 5A 0B 43 59 0A 43 59 0A 43 59
00002AC0:0A 43 59 0A 43 5A 0A 00 40 5D 0D 40 5E 0E 41 5E
00002AD0:0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D
00002AE0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F
00002AF0:5C 0C 3F 5C 0C 3F 5C 0C 3F 5C 0C 40 5D 0D 40 5D
00002B00:0D 40 5D 0D 3F 5C 0C 3F 5C 0C 44 60 13 58 70 2E
00002B10:73 85 53 84 92 6B 8F 99 7C A6 AC 9A B7 B8 AE B9
00002B20:B9 B8 BB BD BE BA BD BF B8 BB BD B5 B8 BB B0 B3
00002B30:B6 A9 AB AD 9D 9E 9F 8F 8F 90 86 86 86 87 86 86
00002B40:B5 B1 AD DE D8 D1 BB B5 AF 6C 6C 6C 79 79 79 6F
00002B50:6F 6F 71 72 71 7A 7C 76 7D 83 73 7C 86 68 7B 86
00002B60:60 6F 7E 4D 57 6B 2A 46 5E 11 43 5C 0C 43 5D 0A
00002B70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42
00002B80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B
00002B90:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5A 0B 42 5A 0B
00002BA0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43
00002BB0:5C 0A 43 5C 0A 43 5C 0A 44 5B 09 44 5A 0A 44 5A
00002BC0:0A 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 59 0A
00002BD0:44 5A 0B 00 3E 5E 0D 3F 5D 0D 3F 5D 0D 40 5E 0E
00002BE0:41 5E 0E 40 5D 0D 40 5D 0D 3F 5C 0C 40 5D 0C 40
00002BF0:5D 0C 40 5D 0C 40 5D 0C 3F 5C 0B 3F 5C 0B 3F 5C
00002C00:0B 3F 5C 0B 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C
00002C10:3F 5C 0C 3F 5C 0C 40 5D 0D 40 5D 0D 40 5D 0D 40
00002C20:5D 0D 40 5D 0D 41 5C 0C 48 62 16 52 69 23 58 6E
00002C30:2D 5D 72 34 61 75 3A 64 77 3F 67 79 42 68 7A 44
00002C40:69 7B 46 69 7A 46 68 78 44 66 77 42 64 75 40 8C
00002C50:96 6D C0 BE B0 51 62 2B 57 6B 2D 58 6D 2C 51 67
00002C60:22 49 60 15 41 5B 0B 41 5B 0B 41 5B 0B 41 5B 0B
00002C70:41 5B 0B 40 5B 0B 40 5B 0B 40 5C 09 42 5B 0A 42
00002C80:5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B
00002C90:0A 41 5A 0A 42 5A 0C 42 5A 0C 42 5A 0C 42 5B 0B
00002CA0:42 5B 0B 42 5B 0B 42 5B 0A 42 5B 0A 44 5A 0B 44
00002CB0:5A 0B 44 5A 0A 44 5A 0A 44 5A 0A 45 5B 0B 45 5B
00002CC0:0B 45 5B 0B 44 5A 09 44 5A 09 44 5A 0B 44 5A 0B
00002CD0:44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 5B 09 00
00002CE0:3F 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F 0D 40 5E 0D 40
00002CF0:5E 0D 40 5E 0D 40 5E 0D 40 5F 0C 40 5F 0C 40 5F
00002D00:0C 40 5F 0C 3F 5E 0B 3F 5E 0B 40 5E 0B 40 5E 0B
```

```
00002D10:40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40
00002D20:5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D
00002D30:0C 41 5D 0C 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B
00002D40:41 5D 0C 41 5D 0C 41 5D 0C 40 5C 0B 40 5C 0B 40
00002D50:5C 0B 40 5C 0B 40 5C 0C 41 5C 0C 41 5C 0C 5A 70
00002D60:2B 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C
00002D70:41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41
00002D80:5C 0B 41 5C 0B 41 5D 0B 42 5D 0A 42 5D 0A 42 5D
00002D90:0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A
00002DA0:42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 43
00002DB0:5C 0B 43 5D 0A 43 5D 0A 44 5C 0B 44 5C 0B 44 5C
00002DC0:0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A
00002DD0:43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43
00002DE0:5B 0A 43 5B 0A 43 5B 0A 43 5D 09 00 3F 61 0C 40
00002DF0:62 0D 40 62 0D 3F 61 0C 3F 61 0C 3F 61 0C 3F 61
00002E00:0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C
00002E10:3F 60 0C 3F 60 0C 40 60 0C 40 60 0C 40 60 0C 40
00002E20:60 0C 3F 5F 0C 3F 5F 0C 3F 5F 0C 40 60 0C 40 60
00002E30:0C 40 60 0C 3F 5F 0B 3F 5F 0B 3F 5F 0C 41 5F 0C
00002E40:41 5E 0B 41 5E 0B 42 5F 0C 42 5F 0C 40 60 0C 40
00002E50:60 0C 40 60 0C 3F 5F 0B 3F 5F 0B 3F 5F 0B 3F 5F
00002E60:0B 3F 5F 0B 42 5F 0C 42 5F 0C 41 5E 0B 41 5E 0B
00002E70:41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0C 41 5E 0B 41
00002E80:5E 0B 41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0B 41 5F
00002E90:0B 41 5F 0A 41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B
00002EA0:41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 42 5F 0B 42
00002EB0:5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 43 5F 0B 43 5E
00002EC0:0A 43 5E 0A 43 5F 0B 43 5F 0B 43 5F 0B 43 5F 0B
00002ED0:43 5F 0B 43 5F 0A 43 5F 0A 43 5F 0A 42 5E 09 42
00002EE0:5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E
00002EF0:09 42 5E 09 42 5E 09 00 3F 64 0B 3F 64 0B 3F 64
00002F00:0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B
00002F10:3F 64 0C 3F 64 0C 3F 64 0C 3F 62 0C 40 62 0C 40
00002F20:62 0C 40 62 0C 40 62 0C 40 62 0C 40 62 0C 3F 62
00002F30:0C 3F 62 0C 40 63 0C 40 63 0D 40 62 0C 40 62 0C
00002F40:3F 62 0B 3F 62 0B 3F 62 0C 40 62 0C 40 62 0C 41
00002F50:62 0C 41 63 0D 40 62 0C 40 63 0A 40 63 0A 40 63
00002F60:0A 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C
00002F70:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40
00002F80:61 0B 40 61 0B 41 62 0C 40 61 0B 40 61 0B 40 61
00002F90:0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B
00002FA0:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40
00002FB0:61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61
00002FC0:0B 40 62 09 40 62 09 40 62 09 42 61 09 42 61 09
00002FD0:42 61 09 42 61 09 42 61 09 42 61 09 42 61 09 42
00002FE0:61 09 42 61 09 42 61 09 40 62 09 42 61 09 42 61
00002FF0:09 42 61 09 42 61 09 42 61 09 42 61 09 42 61 09
00003000:42 61 09 00 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F
00003010:3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D
00003020:60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60
00003030:0F 3D 60 0F 3D 61 0E 3D 61 0E 3C 60 0D 3C 60 0D
00003040:3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D
00003050:61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61
00003060:0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E
00003070:3C 60 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E
00003080:5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F
00003090:0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D
000030A0:3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E
000030B0:5F 0D 3E 5F 0D 3E 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F
000030C0:0D 3F 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D
000030D0:3E 60 0B 3D 5F 0A 3E 5F 0A 3E 5F 0A 3F 60 0B 3F
000030E0:60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60
000030F0:0B 3F 60 0B 3E 60 0B 3E 60 0B 3E 60 0B 3F 60 0B
00003100:3F 60 0B 3F 60 0B 3F 5F 0D 3F 5F 0D 3D 5F 0A 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x0000004C)																															
Size (0x00002FA8)																															
Bounds (0x00000000)																															
... (0x0000002D)																															
... (0x00000059)																															
... (0x00000059)																															
xDest (0x00000000)																															
yDest (0x0000002D)																															
cxDest (0x0000005A)																															
cyDest (0x0000002D)																															

Type (4 bytes): 0x0000004C identifies this record type as EMR_BITBLT.

Size (4 bytes): 0x00002FA8 specifies the size of this record in bytes.

Bounds (16 bytes): 0x00000000, 0x00000002D, 0x00000059, 0x00000059 specifies the bounding rectangle in logical units.

xDest (4 bytes): 0x00000000 specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

yDest (4 bytes): 0x0000002D specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

cxDest (4 bytes): 0x0000005A specifies the logical width of the destination rectangle.

cyDest (4 bytes): 0x0000002D specifies the logical height of the destination rectangle.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BitBlitRasterOperation (0x00CC0020)																															
xSrc (0x00000000)																															
ySrc (0x00000000)																															
xformSrc (0x3F800000) (24 bytes)																															
... (0x00000000)																															

... (0x00000000)
... (0x3F800000)
... (0x00000000)
... (0x00000000)

BitBlitRasterOperation (4 bytes): 0x00CC0020 specifies the raster operation code from the **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 (4 bytes): 0x00000000 specifies the logical x-coordinate of the upper-left corner of the source rectangle.

ySrc (4 bytes): 0x00000000 specifies the logical y-coordinate of the upper-left corner of the source rectangle.

xformSrc (24 bytes): 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BkColorSrc (0x00FFFFFF)																															
UsageSrc (0x00000000)																															
offBmiSrc (0x00000064)																															
cbBmiSrc (0x00000000)																															
offBitsSrc (0x0000008C)																															
cbBitsSrc (0x00002F1C)																															

BkColorSrc (4 bytes): 0x00FFFFFF specifies the background RGB color.

UsageSrc (4 bytes): 0x00000000 specifies the value of the **Colors** field of the **DeviceIndependentBitmap** object ([MS-WMF] section 2.2.2.9) from the DIBColors enumeration (section 2.1.9).

offBmiSrc (4 bytes): 0x00000064 specifies the offset to the source **DeviceIndependentBitmap** object.

cbBmiSrc (4 bytes): 0x00000000 specifies the size of the source **DeviceIndependentBitmap** object.

offBitsSrc (4 bytes): 0x0000008C specifies the offset to the source bitmap bits.

cbBitsSrc (4 bytes): 0x00002F1C specifies the size of the source bitmap bits.

3.2.7 EMR_SETBKMODE Example

This section provides an example of an EMR_SETBKMODE record (section 2.3.11.11).

```
00003110:12 00 00 00 0C 00 00 00 01 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000012)																															
Size (0x0000000C)																															
Mode (0x00000001)																															

Type (4 bytes): 0x00000012 identifies this record type as EMR_SETBKMODE.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

Mode (4 bytes): 0x00000001 specifies the background color value.

3.2.8 EMR_EXTCREATEFONTINDIRECTW Example 1

This section provides an example of an 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
00003130:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 01
00003140:04 00 00 02 41 00 72 00 69 00 61 00 6C 00 00 00
00003150:00 00 00 00 00 00 00 00 00 00 00 00 0C 45 00 00
00003160:BC 16 E8 FE FE 07 00 00 20 00 CC 00 00 00 00 00
00003170:4C 00 00 00 FE 07 00 00 DA 16 01 B8 FF FF FF FF
00003180:00 00 00 00 00 00 00 00 F0 F6 13 00 00 00 00 00
00003190:0E 20 05 27 00 00 00 00 28 00 00 00 00 00 00 00
000031A0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031B0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031C0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031D0:00 00 00 00 00 00 00 00 28 00 00 00 FF FF FF 00
000031E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031F0:58 00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00
00003200:00 00 80 3F 00 00 00 00 00 00 00 00 00 00 80 3F
00003210:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003220:D0 C5 35 00 00 00 00 00 28 00 00 00 59 00 00 00
00003230:2D 00 00 00 01 00 18 00 00 00 00 00 1C 2F 00 00
00003240:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003250:10 00 90 01 00 00 00 00 25 00 00 00 00 00 00 00
00003260:D3 3F EC FE FE 07 00 00 79 0D 21 11 00 00 00 00
00003270:10 00 90 01 00 00 00 00 00 00 00 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000052)																															

Size (0x00000170)
ihFonts (0x00000002)
elw (360 bytes)
...
...
...
...

Type (4 bytes): 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size (4 bytes): 0x00000170 specifies the size of this record in bytes.

ihFonts (4 bytes): 0x00000002 specifies the object index that is assigned to the font.

elw (360 bytes): 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 LogFontExDv object (section 2.2.15).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
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") (68 bytes)																															
...																															
...																															
...																															

Height (4 bytes): 0xFFFFFFF3 has an absolute value of 13, which specifies the character height for this font, in logical units.

Width (4 bytes): 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 (4 bytes): 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

Orientation (4 bytes): 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

Weight (4 bytes): 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 (1 byte): 0x00 specifies that the font is not italic.

Underline (1 byte): 0x00 specifies that the font is not underlined.

Strikeout (1 byte): 0x00 specifies that the font characters do not have a strike-out graphic.

CharSet (1 byte): 0x01 specifies the default character set, from the CharacterSet enumeration ([MS-WMF] section 2.1.1.5).

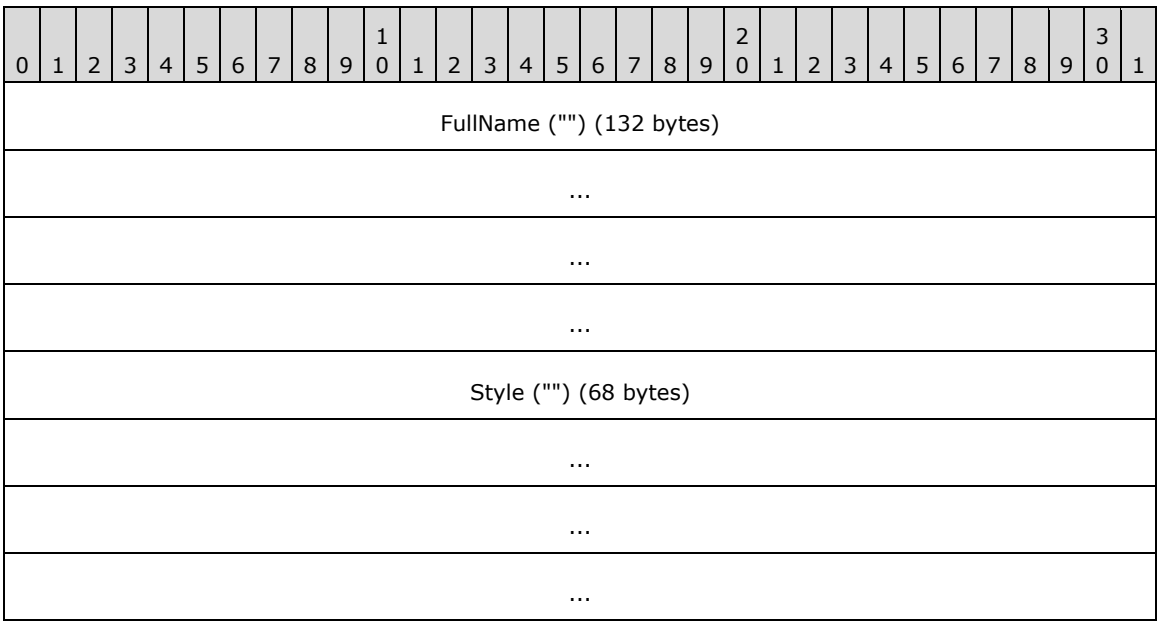
OutPrecision (1 byte): 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the OutPrecision enumeration ([MS-WMF] section 2.1.1.21). The value 0x04 specifies a TrueType font, if there is a choice between multiple fonts with the same name.

ClipPrecision (1 byte): 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping regions, from the ClipPrecision flags ([MS-WMF] section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality (1 byte): 0x00 specifies default output quality, from the FontQuality enumeration ([MS-WMF] section 2.1.1.10).

PitchAndFamily (1 byte): 0x02 specifies a variable-pitch font, and no preference for font family, from the PitchAndFamily object ([MS-WMF] section 2.2.2.14).

Facename (68 bytes): "Arial" specifies the typeface name of the font in Unicode characters.



Script ("") (68 bytes)
...
...
...
Signature (0x08007664)
NumAxes (0x00000000)

FullName (132 bytes): An empty string specifies the font's full name.

Style (68 bytes): An empty string describes the font's style.

Script (68 bytes): An empty string describes the font's character set.

Signature (4 bytes): 0x08007664 specifies the signature of a DesignVector object (section 2.2.3).

NumAxes (4 bytes): 0x00000000 specifies the number of font axes described in the DesignVector object.

3.2.9 EMR_SELECTOBJECT Example 3

This section provides an example of an 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000002)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000002 specifies the index of an object in the EMF object table.

3.2.10 EMR_EXTTEXTOUTW Example

This section provides an example of an 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 00 FF FF FF FF FF FF FF FF
000032B0:01 00 00 00 AB 0A 0D 42 00 00 0D 42 12 00 00 00
```

```
000032C0:05 00 00 00 0E 00 00 00 4C 00 00 00 00 00 00 00
000032D0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF
000032E0:68 00 00 00 53 00 69 00 6D 00 70 00 6C 00 65 00
000032F0:20 00 53 00 61 00 6D 00 70 00 6C 00 65 00 00 00
00003300:09 00 00 00 03 00 00 00 0B 00 00 00 07 00 00 00
00003310:03 00 00 00 07 00 00 00 04 00 00 00 09 00 00 00
00003320:07 00 00 00 0B 00 00 00 07 00 00 00 03 00 00 00
00003330:07 00 00 00 09 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000054)																															
Size (0x000000A0)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0xFFFFFFFF)																															
... (0xFFFFFFFF)																															
iGraphicsMode (0x00000001)																															
exScale (35.260418)																															
eyScale (35.250000)																															
aemrtext (variable)																															
...																															

Type (4 bytes): 0x00000054 identifies this record type as EMR_EXTTEXTOUTW.

Size (4 bytes): 0x000000A0 specifies the size of this record in bytes.

Bounds (16 bytes): 0x00000000, 0x00000000, 0xFFFFFFFF, 0xFFFFFFFF values are not used.

iGraphicsMode (4 bytes): 0x00000001 specifies the current graphics mode.

exScale (4 bytes): 35.260418 specifies the X scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

eyScale (4 bytes): 35.250000 specifies the Y scale from page units to .01mm units if graphics mode is **GM_COMPATIBLE**.

aemrtext (variable): 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Reference (0x00000012)																															

... (0x00000005)
Chars (0x0000000E)
offString (0x0000004C)
Options (0x00000000)
Rectangle (0x00000000)
... (0x00000000)
... (0xFFFFFFFF)
... (0xFFFFFFFF)
offDx (0x00000068)
text ("Simple Sample")

Reference (8 bytes): 0x00000012, 0x00000005 specify the coordinates of the reference point used to position the string.

Chars (4 bytes): 0x0000000E specifies the number of characters in the string.

offString (4 bytes): 0x0000004C specifies the offset to the string in bytes, from the start of the EMR_EXTTEXTOUTW record.

Options (4 bytes): 0x00000000 specifies that the **Rectangle** field is not used.

Rectangle (16 bytes): 0x00000000, 0x00000000, 0xFFFFFFFF, 0xFFFFFFFF values are not used.

offDx (4 bytes): 0x00000068 specifies the offset to an intercharacter spacing array in bytes, from the start of the EMR_EXTTEXTOUTW record.

text (4 bytes): "Simple Sample".

3.2.11 EMR_EXTCREATEFONTINDIRECTW Example 2

This section provides an example of an 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 10 00 90 01
000033D0:00 00 00 00 25 00 00 00 00 00 00 00 D3 3F EC FE
000033E0:FE 07 00 00 79 0D 21 11 00 00 00 00 10 00 90 01
000033F0:00 00 00 00 00 00 00 00 59 00 00 00 2D 00 00 00
00003400:64 76 00 08 00 00 00 00 00 00 00 00 00 08 0D 00
00003410:0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003420:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00003430:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003440:00 00 00 00 92 A0 CD 02 00 00 00 00 CA BE CD 02
00003450:00 00 00 00 00 00 00 00 00 00 00 00 FF FF 5A FE
00003460:00 00 00 00 00 00 00 00 00 00 00 00 95 F1 53 FE
00003470:FE 07 00 00 BE 06 5A FE FE 07 00 00 87 F2 53 FE
00003480:FE 07 00 00 C4 04 5A FE FE 07 00 00 79 0D 21 11
00003490:00 00 00 00 01 00 00 00 00 00 00 00 F0 02 5A FE
000034A0:64 76 00 08 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000003)																															
elw (360 bytes)																															
...																															
...																															
...																															
...																															

Type (4 bytes): 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size (4 bytes): 0x00000170 specifies the size of this record in bytes.

ihFonts (4 bytes): 0x00000003 specifies the object index that is assigned to the font.

elw (360 bytes): 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 LogFontExDv object (section 2.2.15).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Height (0xFFFFFFF3)																															
Width (0x00000000)																															
Escapement (0x00000C4E)																															
Orientation (0x00000C4E)																															
Weight 0x000000C8)																															
Italic (0x00)									Underline (0x00)										StrikeOut (0x00)										CharSet (0x00)							
OutPrecision (0x04)									ClipPrecision (0x00)										Quality (0x00)										PitchAndFamily (0x02)							

Facename ("Microsoft Sans Serif") (68 bytes)
...
...
...

Height (4 bytes): 0xFFFFFFF3 has an absolute value of 13, which specifies the character height for this font in logical units.

Width (4 bytes): 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 (4 bytes): 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

Orientation (4 bytes): 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

Weight (4 bytes): 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 (1 byte): 0x00 specifies that the font is not italic.

Underline (1 byte): 0x00 specifies that the font is not underlined.

Strikeout (1 byte): 0x00 specifies that the font characters do not have a strike-out graphic.

CharSet (1 byte): 0x00 specifies the **ANSI_CHARSET**, as defined in the CharacterSet enumeration ([MS-WMF] section 2.1.1.5).

OutPrecision (1 byte): 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the 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 (1 byte): 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the ClipPrecision flags ([MS-WMF] section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality (1 byte): 0x00 specifies default output quality, from the FontQuality enumeration ([MS-WMF] section 2.1.1.10).

PitchAndFamily (1 byte): 0x02 specifies a variable-pitch font, and no preference for font family, from the PitchAndFamily object ([MS-WMF] section 2.2.2.14).

Facename (68 bytes): "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
FullName ("") (132 bytes)																															
...																															

...
...
Style ("") (68 bytes)
...
...
...
Script ("") (68 bytes)
...
...
...
Signature (0x80007664)
NumAxes (0x00000000)

FullName (132 bytes): An empty string specifies the font's full name.

Style (68 bytes): An empty string describes the font's style.

Script (68 bytes): An empty string describes the font's character set.

Signature (4 bytes): 0x08007664 specifies the signature of a DesignVector object (section 2.2.3).

NumAxes (4 bytes): 0x00000000 specifies the number of font axes described in the DesignVector object.

3.2.12 EMR_SELECTOBJECT Example 4

This section provides an example of an EMR_SELECTOBJECT record (section 2.3.8.5).

```
000034A0:                       25 00 00 00 0C 00 00 00
000034B0:03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

Type (4 bytes): Identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): Specifies the size of this record in bytes.

ihObject (4 bytes): Specifies the index of an object in the EMF object table.

3.2.13 EMR_EXTCREATEFONTINDIRECTW Example 3

This section provides an example of an 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 00 08 00 00 3B 09 00 00 00 00 00 00
00003570:A0 E8 07 02 00 00 00 00 03 00 00 00 FE 07 00 00
00003580:90 01 00 00 4D 00 69 00 63 00 72 00 6F 00 73 00
00003590:6F 00 66 00 74 00 20 00 53 00 61 00 00 00 73 00
000035A0:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00
000035B0:00 00 00 00 00 00 00 00 00 00 FF FF 5A FE 00 00 00 00
000035C0:40 02 5A FE FE 07 00 00 9D 04 00 00 00 00 00 00
000035D0:FF FF FF FF FF FF FF FF 01 00 00 00 00 00 00 00
000035E0:20 AC CF 02 00 00 00 00 00 00 07 02 00 00 00 00
000035F0:10 AC CF 02 00 00 00 00 26 36 E3 76 00 00 00 00
00003600:00 00 07 02 00 00 00 00 01 00 00 00 00 27 00 00
00003610:00 00 00 00 00 00 00 00 20 AC CF 02 64 76 00 08
00003620:00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000004)																															
elw (360 bytes)																															
...																															
...																															
...																															
...																															

Type (4 bytes): 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size (4 bytes): 0x00000170 specifies the size of this record in bytes.

ihFonts (4 bytes): 0x00000004 specifies the object index that is assigned to the font.

elw (360 bytes): 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 LogFontExDv object (section 2.2.15).

<table>
<tr><td colspan="32">0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1</td></tr>
<tr><td colspan="32">Height (0xFFFFFFF2)</td></tr>
<tr><td colspan="32">Width (0x00000000)</td></tr>
<tr><td colspan="32">Escapement (0x00000C4E)</td></tr>
<tr><td colspan="32">Orientation (0x00000C4E)</td></tr>
<tr><td colspan="32">Weight 0x000000C8)</td></tr>
<tr><td colspan="8">Italic (0x00)</td><td colspan="8">Underline (0x00)</td><td colspan="8">StrikeOut (0x00)</td><td colspan="8">CharSet (0x00)</td></tr>
<tr><td colspan="8">OutPrecision (0x04)</td><td colspan="8">ClipPrecision (0x00)</td><td colspan="8">Quality (0x00)</td><td colspan="8">PitchAndFamily (0x02)</td></tr>
<tr><td colspan="32">Facename ("Microsoft Sans Serif") (68 bytes)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Height (4 bytes): 0xFFFFFFF2 has an absolute value of 14, which specifies the character height for this font in logical units.

Width (4 bytes): 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 (4 bytes): 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

Orientation (4 bytes): 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

Weight (4 bytes): 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 (1 byte): 0x00 specifies that the font is not italic.

Underline (1 byte): 0x00 specifies that the font is not underlined.

Strikeout (1 byte): 0x00 specifies that the font characters do not have a strike-out graphic.

CharSet (1 byte): 0x00 specifies the **ANSI_CHARSET**, as defined in the CharacterSet enumeration ([MS-WMF] section 2.1.1.5).

OutPrecision (1 byte): 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the 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 (1 byte): 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the ClipPrecision flags ([MS-WMF] section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality (1 byte): 0x00 specifies default output quality, from the FontQuality enumeration ([MS-WMF] section 2.1.1.10).

PitchAndFamily (1 byte): 0x02 specifies a variable-pitch font, and no preference for font family, from the PitchAndFamily object ([MS-WMF] section 2.2.2.14).

Facename (68 bytes): "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
FullName ("") (132 bytes)																															
...																															
...																															
...																															
Style ("") (68 bytes)																															
...																															
...																															
...																															
Script ("") (68 bytes)																															
...																															
...																															
...																															
Signature (0x80007664)																															
NumAxes (0x00000000)																															

FullName (132 bytes): An empty string specifies the font's full name.

Style (68 bytes): An empty string describes the font's style.

Script (68 bytes): An empty string describes the font's character set.

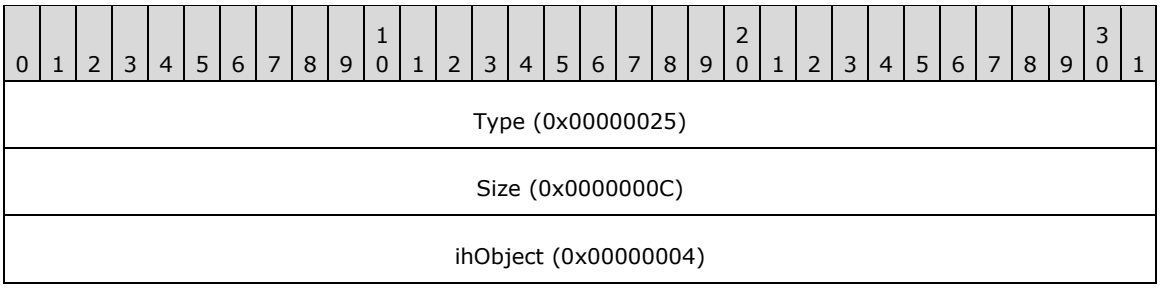
Signature (4 bytes): 0x08007664 specifies the signature of a DesignVector object (section 2.2.3).

NumAxes (4 bytes): 0x00000000 specifies the number of font axes described in the DesignVector object.

3.2.14 EMR_SELECTOBJECT Example 5

This section provides an example of an EMR_SELECTOBJECT record (section 2.3.8.5).

```
00003620:            25 00 00 00 0C 00 00 00 04 00 00 00
```



Type (4 bytes): Identifies this record type as EMR_SELECTOBJECT.

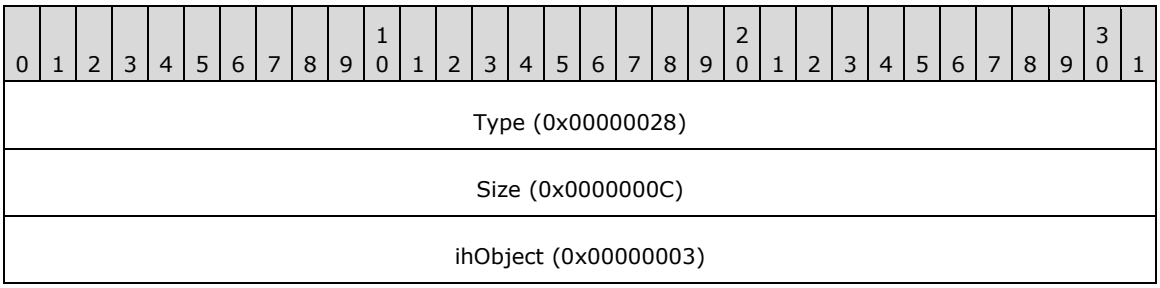
Size (4 bytes): Specifies the size of this record in bytes.

ihObject (4 bytes): Specifies the index of an object in the EMF object table.

3.2.15 EMR_DELETEOBJECT Example 1

This section provides an example of an EMR_DELETEOBJECT record (section 2.3.8.3).

```
00003630:28 00 00 00 0C 00 00 00 03 00 00 00
```



Type (4 bytes): Identifies this record type as EMR_DELETEOBJECT.

Size (4 bytes): Specifies the size of this record in bytes.

ihObject (4 bytes): Specifies the index of the object to be deleted.

3.2.16 EMR_EXTCREATEFONTINDIRECTW Example 4

This section provides an example of an 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
00003660:04 00 00 02 4D 00 69 00 63 00 72 00 6F 00 73 00
00003670:6F 00 66 00 74 00 20 00 53 00 61 00 6E 00 73 00
00003680:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00
00003690:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000036A0:00 00 00 00 00 00 00 73 00 20 00 53 00 65 00 72 00
000036B0:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00
000036C0:FF FF 5A FE 00 00 00 00 40 02 5A FE FE 07 00 00
000036D0:9D 04 00 00 00 00 00 00 FF FF FF FF FF FF FF FF
000036E0:01 00 00 00 00 00 00 00 20 AC CF 02 00 00 00 00
000036F0:00 00 07 02 00 00 00 00 40 02 5A FE FE 07 00 00
00003700:F3 14 00 00 00 00 00 00 F3 14 0A 1E 00 00 00 00
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00
00003720:65 58 53 FE 00 00 00 00 00 00 00 00 00 00 00 00
00003730:00 F5 13 00 00 00 00 00 03 01 56 E5 89 1A 00 00
00003740:55 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00
00003750:00 00 00 00 FE 07 00 00 79 0D 21 11 00 00 00 00
00003760:40 02 5A FE 00 00 00 00 26 06 5A FE FE 07 00 00
00003770:08 F5 13 00 00 00 00 00 00 F5 13 00 00 00 00 00
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00
00003790:04 00 00 00 00 00 00 00 00 24 07 5A FE FE 07 00 00
000037A0:01 00 00 00 64 76 00 08 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000003)																															
elw (360 bytes)																															
...																															
...																															
...																															
...																															

Type (4 bytes): 0x00000052 identifies this record type as EMR_EXTCREATEFONTINDIRECTW.

Size (4 bytes): 0x00000170 specifies the size of this record in bytes.

ihFonts (4 bytes): 0x00000003 specifies the object index that is assigned to the font.

elw (360 bytes): 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 LogFontExDv object (section 2.2.15).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
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") (68 bytes)																															
...																															
...																															
...																															

Height (4 bytes): 0x00000013 specifies the cell height for this font, in logical units.

Width (4 bytes): 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 (4 bytes): 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

Orientation (4 bytes): 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

Weight (4 bytes): 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 (1 byte): 0x00 specifies that the font is not italic.

Underline (1 byte): 0x00 specifies that the font is not underlined.

Strikeout (1 byte): 0x00 specifies that the font characters do not have a strike-out graphic.

CharSet (1 byte): 0x00 specifies the **ANSI_CHARSET**, as defined in the CharacterSet enumeration ([MS-WMF] section 2.1.1.5).

OutPrecision (1 byte): 0x04 specifies the output precision, which is how closely the output is expected to match the requested font properties, from the 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 (1 byte): 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the ClipPrecision flags ([MS-WMF] section 2.1.2.1). The value 0x00 specifies default clipping behavior.

Quality (1 byte): 0x00 specifies default output quality, from the FontQuality enumeration ([MS-WMF] section 2.1.1.10).

PitchAndFamily (1 byte): 0x02 specifies a variable-pitch font, and no preference for font family, from the PitchAndFamily object ([MS-WMF] section 2.2.2.14).

Facename (68 bytes): "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
FullName ("") (132 bytes)																															
...																															
...																															
...																															
Style ("") (68 bytes)																															
...																															
...																															
...																															
Script ("") (68 bytes)																															
...																															
...																															
...																															
Signature (0x80007664)																															
NumAxes (0x00000000)																															

FullName (132 bytes): An empty string specifies the font's full name.

Style (68 bytes): An empty string describes the font's style.

Script (68 bytes): An empty string describes the font's character set.

Signature (4 bytes): 0x08007664 specifies the signature of a DesignVector object (section 2.2.3).

NumAxes (4 bytes): 0x00000000 specifies the number of font axes described in the DesignVector object.

3.2.17 EMR_SELECTOBJECT Example 6

This section provides an example of an 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000003 specifies the index of an object in the EMF object table.

3.2.18 EMR_SELECTOBJECT Example 7

This section provides an example of an EMR_SELECTOBJECT record (section 2.3.8.5).

```
000037B0:                         25 00 00 00 0C 00 00 00
000037C0:02 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000002)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000002 specifies the index of an object in the EMF object table.

3.2.19 EMR_DELETEOBJECT Example 2

This section provides an example of an EMR_DELETEOBJECT record (section 2.3.8.3).

```
000037C0:            28 00 00 00 0C 00 00 00 04 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000028)																															
Size (0x0000000C)																															
ihObject (0x00000004)																															

Type (4 bytes): 0x00000028 identifies this record type as EMR_DELETEOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000004 specifies the index of the object to be deleted.

3.2.20 EMR_DELETEOBJECT Example 3

This section provides an example of an EMR_DELETEOBJECT record (section 2.3.8.3).

```
000037D0:28 00 00 00 0C 00 00 00 03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000028)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

Type (4 bytes): 0x00000028 identifies this record type as EMR_DELETEOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x00000003 specifies the index of the object to be deleted.

3.2.21 EMR_SELECTOBJECT Example 8

This section provides an example of an 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	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x8000000D = SYSTEM_FONT)																															

Type (4 bytes): 0x00000025 identifies this record type as EMR_SELECTOBJECT.

Size (4 bytes): 0x0000000C specifies the size of this record in bytes.

ihObject (4 bytes): 0x8000000D specifies the index of an object in the EMF object table.

3.2.22 EMR_EOF Example

This section provides an example of an 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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x0000000E)																															
Size (0x00000014)																															
nPalEntries (0x00000000)																															
offPalEntries (0x00000010)																															
SizeLast (0x00000014)																															

Type (4 bytes): 0x0000000E identifies this record type as EMR_EOF.

Size (4 bytes): 0x00000014 specifies the size of this record in bytes.

nPalEntries (4 bytes): 0x00000000 specifies the number of palette entries.

offPalEntries (4 bytes): 0x00000010 specifies the offset to the palette entries.

SizeLast (4 bytes): 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~~updates to those products.

- Windows NT 3.1 operating system
- Windows NT 3.5 operating system
- Windows NT 3.51 operating system
- Windows NT 4.0 operating system
- 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
- Windows 10 operating system
- Windows Server 2016 operating system
- Windows Server operating system

Exceptions, if any, are noted ~~below.~~in this section. If ~~a~~ an update version, service pack or ~~Quick Fix Engineering (QFE~~Knowledge Base (KB) number appears with ~~the~~a product ~~version,~~name, the behavior changed in that ~~service pack or QFE.~~update. The new behavior also applies to subsequent ~~service packs of the product~~updates 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 1.3.1: **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.

<2> Section 1.4: Windows applications that use the GDI+ API [MSDN-GDI+] can create EMF metafiles that contain EMF+ records ([MS-EMFPLUS] section 2.3).

<3> 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.

<4> 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.

<5> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_SETICMMODE.

<6> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_CREATECOLORSPACE.

<7> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_SETCOLORSPACE.

<8> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_DELETECOLORSPACE.

<9> Section 2.1.1: Windows uses an EMR_DELETEOBJECT record to delete a logical color space object.

<10> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_GLSRECORD.

<11> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_GLSBOUNDEDRECORD.

<12> Section 2.1.1: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support EMR_PIXELFORMAT.

<13> 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**.

<14> 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**.

<15> 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**.

<16> 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**.

<17> 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**.

<18> 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**.

<19> 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**.

<20> Section 2.1.1: Windows GDI uses an EMR_EXTTEXTOUTW record (section 2.3.5.8) to perform this function.

<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_COLORMATCHTOTARGETW**.

<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_CREATECOLORSPACEW**.

<23> 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 processes the string directly. See [MSDN-GDI+] for more information.

<24> 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.

<25> 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_NUMERICSLOCAL** flag used to indicate the display of numeric digits appropriate to the locale.

<26> 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.

<27> 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 might cause no text to be output.

<28> 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 are provided.

<29> 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.

<30> Section 2.1.16: **GM_COMPATIBLE** graphics mode is used for compatibility between 16-bit and 32-bit systems.

<31> 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.

<32> 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 Image Color Management (ICM).

<33> Section 2.1.31: On Windows, this is the "Courier" font.

<34> Section 2.1.31: On Windows, this is the "MS Sans Serif" font.

<35> Section 2.1.31: On Windows, this is the "Tahoma" font and is 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".

<36> Section 2.1.31: On Windows, this value is considered equivalent to SYSTEM_FONT for the purposes of the screen display of metafiles.

Windows 98 and Windows Millennium Edition: This value is not supported.

<37> Section 2.1.31: On Windows, this palette consists of the static colors in the system palette.

<38> Section 2.1.31: On Windows, the default user interface font is "Tahoma".

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".

<39> 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 brush is undefined.

<40> 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 pen is undefined.

<41> 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
```

<42> Section 2.2.2: Windows 98 and Windows Millennium Edition do not support the ColorAdjustment object.

<43> Section 2.2.2: Windows can generate ColorAdjustment objects with values outside their valid ranges. Such objects are ignored.

<44> Section 2.2.5: In Windows implementations, this is the clipping and/or opaquing rectangle that is passed to GDI methods **ExtTextOutA** and **ExtTextOutW**.

<45> 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.

<46> 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.

<47> 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
Heavy (Black)	900

<48> Section 2.2.22: Windows implementations do not support this flag.

<49> 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.

<50> Section 2.2.22: Windows uses this flag to indicate that the pixel pixel format is supported by GDI.

<51> 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.

<52> Section 2.2.22: Windows does not support alpha bitplanes.

<53> Section 2.2.22: Windows does not support alpha bitplanes.

<54> Section 2.2.22: Windows does not support alpha bitplanes.

<55> Section 2.2.27: In this case, Windows uses the logical font that is currently selected in the playback device context.

<56> 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**.

<57> Section 2.3.1.3: Windows might set this to a non-zero value.

<58> 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**.

<59> 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.

<60> Section 2.3.3.1: Windows NT 3.1, Windows NT 3.51, and Windows NT 4.0 ignore EMR_COMMENT records.

<61> Section 2.3.3.4.3: On playback, the first graphics format recognized by Windows is used to render the image.

<62> Section 2.3.4.2: Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows 98, and Windows Millennium Edition: EMF header extension 1 is not supported.

<63> Section 2.3.4.2: 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 header extension 2 is not supported.

<64> Section 2.3.5.7: Windows GDI emulates EMR_EXTTEXTOUTA with an EMR_EXTTEXTOUTW record.

<65> Section 2.3.5.10: The Windows playback implementation computes the bounding region from the sum of all the rectangles specified by the **RegionData** object in the **RgnData** field.

<66> Section 2.3.5.11: The Windows playback implementation computes the bounding region from the sum of all the rectangles specified by **RgnData** field.

<67> 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.

<68> 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.

<69> Section 2.3.5.14: The Windows playback implementation computes the bounding region from the sum of all the rectangles specified by the **RgnData** field.

<70> 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.

<71> 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.

<72> Section 2.3.7.2: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_CREATECOLORSPACE**.

<73> Section 2.3.7.3: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_CREATECOLORSPACEW**.

<74> 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.

<75> 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**.

<76> Section 2.3.8.2: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_DELETECOLORSPACE**.

<77> 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.

<78> Section 2.3.8.7: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_SETCOLORSPACE**.

<79> 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.

<80> 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**.

<81> Section 2.3.11.1: On Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003, WCS is not enabled in the playback device context before applying the current color transform.

<82> Section 2.3.11.3: The Windows playback implementation computes the bounding rectangle of the region from the sum of all the rectangles specified by the **RegionData** object.

<83> Section 2.3.11.5: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_PIXELFORMAT**.

<84> Section 2.3.11.14: Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 do not support **EMR_SETICMMODE**.

<85> 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**.

<86> 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**.

<87> 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**.

<88> 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**.

<89> Section 2.3.11.21: Windows GDI accepts a FLOAT value for the miter length limit value.

<90> 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 the EMR_SETTEXTJUSTIFICATION record type.

<91> Section 3.1.1.2.1: During metafile playback, before a clipping region is defined an EMF record, Windows uses a default based on the type of device context. in the following table, the referenced functions are described in [MSDN-WindowsGDI] unless stated otherwise.

Device context type	Created by	Default clipping region
Window DC	GetDC	Area of the window passed to GetDC
Memory DC compatible with a specified output device	CreateCompatibleDC with CreateCompatibleBitmap	Area of the compatible bitmap
Monitor DC	MonitorEnumProc from EnumDisplayMonitors	Area of the display monitor
Printer DC	CreateDC	(**dmPelsWidth** X **dmPelsHeight**) of the DEVMODE structure [DEVMODE] input to CreateDC
Metafile DC	CreateMetaFile or CreateEnhMetaFile	None

6 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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.
- A document revision that captures changes to protocol functionality.

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 **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
~~2.3.1 Bitmap Record Types~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~2.3.2 Clipping Record Types~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~2.3.5 Drawing Record Types~~5 Appendix A: Product Behavior	~~7095~~7874 : Added ~~clipping region state information~~Windows Server to applicability list.	~~Minor~~Major
~~2.3.5.10 EMR_FILLRGN Record~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~2.3.5.11 EMR_FRAMERGN Record~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~2.3.5.14 EMR_PAINTRGN Record~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~2.3.11.3 EMR_INVERTRGN Record~~	~~7095 : Added clipping region state information.~~	~~Minor~~
~~3.1.1.2.1 Regions~~	~~7095 : Added default clipping region information.~~	~~Minor~~

7 Index

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X