

libemf

1.0.9

Generated by Doxygen 1.8.17

<b>1 Hierarchical Index</b>	<b>2</b>
1.1 Class Hierarchy	2
<b>2 Data Structure Index</b>	<b>4</b>
2.1 Data Structures	4
<b>3 Data Structure Documentation</b>	<b>8</b>
3.1 EMF::BRUSH Class Reference	8
3.1.1 Detailed Description	8
3.1.2 Constructor & Destructor Documentation	8
3.1.3 Member Function Documentation	9
3.2 EMF::BYTEARRAY Struct Reference	9
3.2.1 Detailed Description	10
3.2.2 Constructor & Destructor Documentation	10
3.3 EMF::CHARSTR Struct Reference	10
3.3.1 Detailed Description	11
3.3.2 Constructor & Destructor Documentation	11
3.4 EMF::DATASTREAM Class Reference	11
3.4.1 Detailed Description	12
3.4.2 Constructor & Destructor Documentation	13
3.4.3 Member Function Documentation	13
3.5 EMF::DWORDARRAY Struct Reference	29
3.5.1 Detailed Description	29
3.5.2 Constructor & Destructor Documentation	30
3.6 EMF::EMRARC Class Reference	30
3.6.1 Detailed Description	30
3.6.2 Constructor & Destructor Documentation	31
3.6.3 Member Function Documentation	31
3.7 EMF::EMRARCTO Class Reference	32
3.7.1 Detailed Description	33
3.7.2 Constructor & Destructor Documentation	33
3.7.3 Member Function Documentation	33
3.8 EMF::EMRBEGINPATH Class Reference	34
3.8.1 Detailed Description	35
3.8.2 Constructor & Destructor Documentation	35
3.8.3 Member Function Documentation	35
3.9 EMF::EMRCLOSEFIGURE Class Reference	36
3.9.1 Detailed Description	37
3.9.2 Constructor & Destructor Documentation	37
3.9.3 Member Function Documentation	37
3.10 EMF::EMRCREATEBRUSHINDIRECT Class Reference	38
3.10.1 Detailed Description	38
3.10.2 Constructor & Destructor Documentation	38

3.10.3 Member Function Documentation	39
3.11 EMF::EMRCREATEPALETTE Class Reference	40
3.11.1 Detailed Description	40
3.11.2 Constructor & Destructor Documentation	40
3.11.3 Member Function Documentation	41
3.12 EMF::EMRCREATEPEN Class Reference	42
3.12.1 Detailed Description	42
3.12.2 Constructor & Destructor Documentation	42
3.12.3 Member Function Documentation	43
3.13 EMF::EMRDELETEOBJECT Class Reference	44
3.13.1 Detailed Description	44
3.13.2 Constructor & Destructor Documentation	44
3.13.3 Member Function Documentation	45
3.14 EMF::EMRELIPSE Class Reference	46
3.14.1 Detailed Description	46
3.14.2 Constructor & Destructor Documentation	46
3.14.3 Member Function Documentation	47
3.15 EMF::EMRENDPATH Class Reference	48
3.15.1 Detailed Description	48
3.15.2 Constructor & Destructor Documentation	48
3.15.3 Member Function Documentation	49
3.16 EMF::EMREOF Class Reference	50
3.16.1 Detailed Description	50
3.16.2 Constructor & Destructor Documentation	50
3.16.3 Member Function Documentation	51
3.17 EMF::EMREXTCREATEFONTINDIRECTW Class Reference	51
3.17.1 Detailed Description	52
3.17.2 Constructor & Destructor Documentation	52
3.17.3 Member Function Documentation	52
3.18 EMF::EMREXTCREATEPEN Class Reference	53
3.18.1 Detailed Description	54
3.18.2 Constructor & Destructor Documentation	54
3.18.3 Member Function Documentation	54
3.19 EMF::EMREXTTEXTOUTA Class Reference	56
3.19.1 Detailed Description	57
3.19.2 Constructor & Destructor Documentation	57
3.19.3 Member Function Documentation	58
3.20 EMF::EMREXTTEXTOUTW Class Reference	59
3.20.1 Detailed Description	59
3.20.2 Constructor & Destructor Documentation	59
3.20.3 Member Function Documentation	60
3.21 EMF::EMRFILLPATH Class Reference	61

---

3.21.1 Detailed Description	61
3.21.2 Constructor & Destructor Documentation	61
3.21.3 Member Function Documentation	62
3.22 EMF::EMRLINETO Class Reference	63
3.22.1 Detailed Description	63
3.22.2 Constructor & Destructor Documentation	63
3.22.3 Member Function Documentation	64
3.23 EMF::EMRMODIFYWORLDTRANSFORM Class Reference	65
3.23.1 Detailed Description	65
3.23.2 Constructor & Destructor Documentation	65
3.23.3 Member Function Documentation	66
3.24 EMF::EMRMOVETOEX Class Reference	67
3.24.1 Detailed Description	67
3.24.2 Constructor & Destructor Documentation	67
3.24.3 Member Function Documentation	68
3.25 EMF::EMRPOLYBEZIER Class Reference	69
3.25.1 Detailed Description	69
3.25.2 Constructor & Destructor Documentation	69
3.25.3 Member Function Documentation	70
3.26 EMF::EMRPOLYBEZIER16 Class Reference	71
3.26.1 Detailed Description	71
3.26.2 Constructor & Destructor Documentation	71
3.26.3 Member Function Documentation	72
3.27 EMF::EMRPOLYBEZIERTO Class Reference	74
3.27.1 Detailed Description	75
3.27.2 Constructor & Destructor Documentation	75
3.27.3 Member Function Documentation	76
3.28 EMF::EMRPOLYBEZIERTO16 Class Reference	76
3.28.1 Detailed Description	77
3.28.2 Constructor & Destructor Documentation	77
3.28.3 Member Function Documentation	78
3.29 EMF::EMRPOLYGON Class Reference	79
3.29.1 Detailed Description	79
3.29.2 Constructor & Destructor Documentation	79
3.29.3 Member Function Documentation	80
3.30 EMF::EMRPOLYGON16 Class Reference	81
3.30.1 Detailed Description	81
3.30.2 Constructor & Destructor Documentation	81
3.30.3 Member Function Documentation	82
3.31 EMF::EMRPOLYLINE Class Reference	84
3.31.1 Detailed Description	85
3.31.2 Constructor & Destructor Documentation	85

3.31.3 Member Function Documentation	86
3.32 EMF::EMRPOLYLINE16 Class Reference	86
3.32.1 Detailed Description	87
3.32.2 Constructor & Destructor Documentation	87
3.32.3 Member Function Documentation	88
3.33 EMF::EMRPOLYLINETO Class Reference	89
3.33.1 Detailed Description	89
3.33.2 Constructor & Destructor Documentation	89
3.33.3 Member Function Documentation	90
3.34 EMF::EMRPOLYLINETO16 Class Reference	91
3.34.1 Detailed Description	91
3.34.2 Constructor & Destructor Documentation	91
3.34.3 Member Function Documentation	92
3.35 EMF::EMRPOLYPOLYGON Class Reference	94
3.35.1 Detailed Description	95
3.35.2 Constructor & Destructor Documentation	95
3.35.3 Member Function Documentation	96
3.36 EMF::EMRPOLYPOLYGON16 Class Reference	96
3.36.1 Detailed Description	97
3.36.2 Constructor & Destructor Documentation	97
3.36.3 Member Function Documentation	98
3.37 EMF::EMRRECTANGLE Class Reference	99
3.37.1 Detailed Description	99
3.37.2 Constructor & Destructor Documentation	99
3.37.3 Member Function Documentation	100
3.38 EMF::EMRRESTOREDC Class Reference	101
3.38.1 Detailed Description	101
3.38.2 Constructor & Destructor Documentation	101
3.38.3 Member Function Documentation	102
3.39 EMF::EMRSAVEDC Class Reference	103
3.39.1 Detailed Description	103
3.39.2 Constructor & Destructor Documentation	103
3.39.3 Member Function Documentation	104
3.40 EMF::EMRSCALEVIEWPORTEXTEX Class Reference	104
3.40.1 Detailed Description	105
3.40.2 Constructor & Destructor Documentation	105
3.40.3 Member Function Documentation	105
3.41 EMF::EMRSCALEWINDOWEXTEx Class Reference	106
3.41.1 Detailed Description	107
3.41.2 Constructor & Destructor Documentation	107
3.41.3 Member Function Documentation	107
3.42 EMF::EMRSELECTOBJECT Class Reference	108

3.42.1 Detailed Description . . . . .	109
3.42.2 Constructor & Destructor Documentation . . . . .	109
3.42.3 Member Function Documentation . . . . .	109
3.43 EMF::EMRSETBKCOLOR Class Reference . . . . .	110
3.43.1 Detailed Description . . . . .	111
3.43.2 Constructor & Destructor Documentation . . . . .	111
3.43.3 Member Function Documentation . . . . .	111
3.44 EMF::EMRSETBKMODE Class Reference . . . . .	112
3.44.1 Detailed Description . . . . .	113
3.44.2 Constructor & Destructor Documentation . . . . .	113
3.44.3 Member Function Documentation . . . . .	113
3.45 EMF::EMRSETMAPMODE Class Reference . . . . .	114
3.45.1 Detailed Description . . . . .	114
3.45.2 Constructor & Destructor Documentation . . . . .	114
3.45.3 Member Function Documentation . . . . .	115
3.46 EMF::EMRSETMETARGN Class Reference . . . . .	116
3.46.1 Detailed Description . . . . .	116
3.46.2 Constructor & Destructor Documentation . . . . .	116
3.46.3 Member Function Documentation . . . . .	117
3.47 EMF::EMRSETMITERLIMIT Class Reference . . . . .	118
3.47.1 Detailed Description . . . . .	118
3.47.2 Constructor & Destructor Documentation . . . . .	118
3.47.3 Member Function Documentation . . . . .	119
3.48 EMF::EMRSETPIXELV Class Reference . . . . .	120
3.48.1 Detailed Description . . . . .	120
3.48.2 Constructor & Destructor Documentation . . . . .	120
3.48.3 Member Function Documentation . . . . .	122
3.49 EMF::EMRSETPOLYFILLMODE Class Reference . . . . .	123
3.49.1 Detailed Description . . . . .	123
3.49.2 Constructor & Destructor Documentation . . . . .	123
3.49.3 Member Function Documentation . . . . .	124
3.50 EMF::EMRSETTEXTALIGN Class Reference . . . . .	125
3.50.1 Detailed Description . . . . .	125
3.50.2 Constructor & Destructor Documentation . . . . .	125
3.50.3 Member Function Documentation . . . . .	126
3.51 EMF::EMRSETTEXTCOLOR Class Reference . . . . .	127
3.51.1 Detailed Description . . . . .	127
3.51.2 Constructor & Destructor Documentation . . . . .	127
3.51.3 Member Function Documentation . . . . .	128
3.52 EMF::EMRSETVIEWPORTEXTEX Class Reference . . . . .	129
3.52.1 Detailed Description . . . . .	129
3.52.2 Constructor & Destructor Documentation . . . . .	129

3.52.3 Member Function Documentation . . . . .	130
3.53 EMF::EMRSETVIEWPORTORGEX Class Reference . . . . .	131
3.53.1 Detailed Description . . . . .	131
3.53.2 Constructor & Destructor Documentation . . . . .	131
3.53.3 Member Function Documentation . . . . .	132
3.54 EMF::EMRSETWINDOWEXTTEX Class Reference . . . . .	133
3.54.1 Detailed Description . . . . .	133
3.54.2 Constructor & Destructor Documentation . . . . .	133
3.54.3 Member Function Documentation . . . . .	134
3.55 EMF::EMRSETWINDOWORGEX Class Reference . . . . .	135
3.55.1 Detailed Description . . . . .	135
3.55.2 Constructor & Destructor Documentation . . . . .	135
3.55.3 Member Function Documentation . . . . .	137
3.56 EMF::EMRSETWORLDTRANSFORM Class Reference . . . . .	138
3.56.1 Detailed Description . . . . .	138
3.56.2 Constructor & Destructor Documentation . . . . .	138
3.56.3 Member Function Documentation . . . . .	139
3.57 EMF::EMRSTROKEANDFILLPATH Class Reference . . . . .	140
3.57.1 Detailed Description . . . . .	140
3.57.2 Constructor & Destructor Documentation . . . . .	140
3.57.3 Member Function Documentation . . . . .	141
3.58 EMF::EMRSTROKEPATH Class Reference . . . . .	142
3.58.1 Detailed Description . . . . .	142
3.58.2 Constructor & Destructor Documentation . . . . .	142
3.58.3 Member Function Documentation . . . . .	143
3.59 EMF::ENHMETAHEADER Class Reference . . . . .	144
3.59.1 Detailed Description . . . . .	144
3.59.2 Constructor & Destructor Documentation . . . . .	144
3.59.3 Member Function Documentation . . . . .	145
3.60 EMF::EXTPEN Class Reference . . . . .	146
3.60.1 Detailed Description . . . . .	146
3.60.2 Constructor & Destructor Documentation . . . . .	146
3.60.3 Member Function Documentation . . . . .	146
3.61 EMF::FONT Class Reference . . . . .	147
3.61.1 Detailed Description . . . . .	148
3.61.2 Constructor & Destructor Documentation . . . . .	148
3.61.3 Member Function Documentation . . . . .	148
3.62 EMF::GLOBALOBJECTS Class Reference . . . . .	149
3.62.1 Detailed Description . . . . .	151
3.62.2 Member Function Documentation . . . . .	151
3.63 EMF::GRAPHICSOBJECT Class Reference . . . . .	154
3.63.1 Detailed Description . . . . .	154

3.63.2 Member Function Documentation	154
3.63.3 Field Documentation	155
3.64 EMF::INTARRAY Struct Reference	155
3.64.1 Detailed Description	155
3.64.2 Constructor & Destructor Documentation	155
3.65 EMF::METAFILEDEVICECONTEXT Class Reference	156
3.65.1 Detailed Description	157
3.65.2 Constructor & Destructor Documentation	157
3.65.3 Member Function Documentation	158
3.65.4 Field Documentation	160
3.66 EMF::METARECORD Class Reference	161
3.66.1 Detailed Description	161
3.66.2 Constructor & Destructor Documentation	162
3.66.3 Member Function Documentation	162
3.67 EMF::OBJECT Class Reference	163
3.67.1 Detailed Description	164
3.67.2 Constructor & Destructor Documentation	164
3.67.3 Member Function Documentation	164
3.67.4 Field Documentation	164
3.68 EMF::PADDING Struct Reference	165
3.68.1 Detailed Description	165
3.68.2 Constructor & Destructor Documentation	165
3.69 EMF::PALETTE Class Reference	166
3.69.1 Detailed Description	166
3.69.2 Constructor & Destructor Documentation	166
3.69.3 Member Function Documentation	167
3.70 EMF::PEN Class Reference	167
3.70.1 Detailed Description	168
3.70.2 Constructor & Destructor Documentation	168
3.70.3 Member Function Documentation	168
3.71 EMF::POINT16ARRAY Struct Reference	169
3.71.1 Detailed Description	169
3.71.2 Constructor & Destructor Documentation	169
3.72 EMF::POINTLARRAY Struct Reference	170
3.72.1 Detailed Description	170
3.72.2 Constructor & Destructor Documentation	170
3.73 EMF::WCHARSTR Struct Reference	171
3.73.1 Detailed Description	171
3.73.2 Constructor & Destructor Documentation	171



# 1 Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<b>EMF::BYTEARRAY</b>	<b>9</b>
<b>EMF::CHARSTR</b>	<b>10</b>
<b>EMF::DATASTREAM</b>	<b>11</b>
<b>EMF::DWORDARRAY</b>	<b>29</b>
<b>EMF::GLOBALOBJECTS</b>	<b>149</b>
<b>EMF::INTARRAY</b>	<b>155</b>
<b>EMF::METARECORD</b>	<b>161</b>
<b>EMF::EMRARC</b>	<b>30</b>
<b>EMF::EMRARCTO</b>	<b>32</b>
<b>EMF::EMRBEGINPATH</b>	<b>34</b>
<b>EMF::EMRCLOSEFIGURE</b>	<b>36</b>
<b>EMF::EMRCREATEBRUSHINDIRECT</b>	<b>38</b>
<b>EMF::EMRCREATEPALETTE</b>	<b>40</b>
<b>EMF::EMRCREATEPEN</b>	<b>42</b>
<b>EMF::EMRDELETEOBJECT</b>	<b>44</b>
<b>EMF::EMRELLIPSE</b>	<b>46</b>
<b>EMF::EMRENDPATH</b>	<b>48</b>
<b>EMF::EMREOF</b>	<b>50</b>
<b>EMF::EMREXTCREATEFONTINDIRECTW</b>	<b>51</b>
<b>EMF::EMREXTCREATEPEN</b>	<b>53</b>
<b>EMF::EMREXTTEXTOUTA</b>	<b>56</b>
<b>EMF::EMREXTTEXTOUTW</b>	<b>59</b>
<b>EMF::EMRFILLPATH</b>	<b>61</b>
<b>EMF::EMRLINETO</b>	<b>63</b>
<b>EMF::EMRMODIFYWORLDTRANSFORM</b>	<b>65</b>
<b>EMF::EMRMOVETOEX</b>	<b>67</b>
<b>EMF::EMRPOLYBEZIER</b>	<b>69</b>
<b>EMF::EMRPOLYBEZIERTO</b>	<b>74</b>

EMF::EMRPOLYBEZIER16	71
EMF::EMRPOLYBEZIERTO16	76
EMF::EMRPOLYBEZIERTO	74
EMF::EMRPOLYBEZIERTO16	76
EMF::EMRPOLYGON	79
EMF::EMRPOLYGON16	81
EMF::EMRPOLYLINE	84
EMF::EMRPOLYLINE16	86
EMF::EMRPOLYLINETO	89
EMF::EMRPOLYLINETO16	91
EMF::EMRPOLYPOLYGON	94
EMF::EMRPOLYPOLYGON16	96
EMF::EMRRECTANGLE	99
EMF::EMRRESTOREDC	101
EMF::EMRSAVEDC	103
EMF::EMRSCALEVIEWPORTEXTEX	104
EMF::EMRSCALEWINDOWEXTEX	106
EMF::EMRSELECTOBJECT	108
EMF::EMRSETBKCOLOR	110
EMF::EMRSETBKMODE	112
EMF::EMRSETMAPMODE	114
EMF::EMRSETMETARGN	116
EMF::EMRSETMITERLIMIT	118
EMF::EMRSETPIXELV	120
EMF::EMRSETPOLYFILLMODE	123
EMF::EMRSETTEXTALIGN	125
EMF::EMRSETTEXTCOLOR	127
EMF::EMRSETVIEWPORTEXTEX	129
EMF::EMRSETVIEWPORTORGEX	131
EMF::EMRSETWINDOWEXTEX	133
EMF::EMRSETWINDOWORGEX	135
EMF::EMRSETWORLDTRANSFORM	138

EMF::EMRSTROKEANDFILLPATH	140
EMF::EMRSTROKEPATH	142
EMF::ENHMETAHEADER	144
EMF::OBJECT	163
EMF::GRAPHICSOBJECT	154
EMF::BRUSH	8
EMF::EXTPEN	146
EMF::FONT	147
EMF::PALETTE	166
EMF::PEN	167
EMF::METAFILEDEVICECONTEXT	156
EMF::PADDING	165
EMF::POINT16ARRAY	169
EMF::POINTLARRAY	170
EMF::WCHARSTR	171

## 2 Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

EMF::BRUSH	
Graphics Brush	8
EMF::BYTEARRAY	
Represent a byte array in a simple way	9
EMF::CHARSTR	
Represent an ASCII character string in a simple way	10
EMF::DATASTREAM	
Support different endian modes when reading and writing the metafile	11
EMF::DWORDARRAY	
Represent an array of double word integers in a simple way	29
EMF::EMRARC	
EMF Arc	30
EMF::EMRARCTO	
EMF Arc To	32
EMF::EMRBEGINPATH	
EMF Begin Path	34

<b>EMF::EMRCLOSEFIGURE</b>	
EMF Close Figure	36
<b>EMF::EMRCREATEBRUSHINDIRECT</b>	
EMF Brush	38
<b>EMF::EMRCREATEPALETTE</b>	
EMF Palette	40
<b>EMF::EMRCREATEPEN</b>	
EMF Pen	42
<b>EMF::EMRDELETEOBJECT</b>	
EMF Delete Object	44
<b>EMF::EMRELLIPSE</b>	
EMF Ellipse	46
<b>EMF::EMRENDPATH</b>	
EMF End Path	48
<b>EMF::EMREOF</b>	
EMF End of File Record	50
<b>EMF::EMREXTCREATEFONTINDIRECTW</b>	
EMF Font	51
<b>EMF::EMREXTCREATEPEN</b>	
EMF Extended Pen	53
<b>EMF::EMREXTTEXTOUTA</b>	
EMF Extended Text Output ASCII	56
<b>EMF::EMREXTTEXTOUTW</b>	
EMF Extended Text Output Wide character	59
<b>EMF::EMRFILLPATH</b>	
EMF Fill path	61
<b>EMF::EMRLINETO</b>	
EMF Line To	63
<b>EMF::EMRMODIFYWORLDTRANSFORM</b>	
EMF Modify World Transform	65
<b>EMF::EMRMOVETOEX</b>	
EMF MoveTo (ex)	67
<b>EMF::EMRPOLYBEZIER</b>	
EMF Polybezier	69
<b>EMF::EMRPOLYBEZIER16</b>	
EMF Polybezier16	71
<b>EMF::EMRPOLYBEZIERTO</b>	
EMF PolyBezierTo	74
<b>EMF::EMRPOLYBEZIERTO16</b>	
EMF PolyBezierTo16	76
<b>EMF::EMRPOLYGON</b>	
EMF Filled Polygon	79

<b>EMF::EMRPOLYGON16</b>	
EMF Filled Polygon16	81
<b>EMF::EMRPOLYLINE</b>	
EMF Polyline	84
<b>EMF::EMRPOLYLINE16</b>	
EMF Polyline16	86
<b>EMF::EMRPOLYLINETO</b>	
EMF PolylineTo	89
<b>EMF::EMRPOLYLINETO16</b>	
EMF PolylineTo16	91
<b>EMF::EMRPOLYPOLYGON</b>	
EMF Poly Polygon	94
<b>EMF::EMRPOLYPOLYGON16</b>	
EMF Poly Polygon16	96
<b>EMF::EMRRECTANGLE</b>	
EMF Rectangle	99
<b>EMF::EMRSTOREDC</b>	
EMF Restore DC	101
<b>EMF::EMRSAVEDC</b>	
EMF Save DC	103
<b>EMF::EMRSCALEVIEWPORTEXT</b>	
EMF Scale Viewport Extents (ex)	104
<b>EMF::EMRSCALEWINDOWEXT</b>	
EMF Scale Window Extents (ex)	106
<b>EMF::EMRSELECTOBJECT</b>	
EMF Select Object	108
<b>EMF::EMRSETBKCOLOR</b>	
EMF Set Background Color	110
<b>EMF::EMRSETBKMODE</b>	
EMF Set Background Mode	112
<b>EMF::EMRSETMAPMODE</b>	
EMF Set Mapping Mode	114
<b>EMF::EMRSETMETARGN</b>	
EMF Set Meta Region	116
<b>EMF::EMRSETMITERLIMIT</b>	
EMF SetMiterLimit	118
<b>EMF::EMRSETPIXELV</b>	
EMF Set Pixel	120
<b>EMF::EMRSETPOLYFILLMODE</b>	
EMF Set the Polygon Fill Mode	123
<b>EMF::EMRSETTEXTALIGN</b>	
EMF Set Text Alignment	125

<b>EMF::EMRSETTEXTCOLOR</b>	
EMF Set Text Color	127
<b>EMF::EMRSETVIEWPORTEXTEX</b>	
EMF Set Viewport Extents (ex)	129
<b>EMF::EMRSETVIEWPORTORGEX</b>	
EMF Set Viewport Origin (ex)	131
<b>EMF::EMRSETWINDOWEXTEX</b>	
EMF Set Window Extent (ex)	133
<b>EMF::EMRSETWINDOWORGEX</b>	
EMF Set Window Origin (ex)	135
<b>EMF::EMRSETWORLDTRANSFORM</b>	
EMF Set World Transform	138
<b>EMF::EMRSTROKEANDFILLPATH</b>	
EMF Stroke and Fill path	140
<b>EMF::EMRSTROKEPATH</b>	
EMF Stroke path	142
<b>EMF::ENHMETAHEADER</b>	
Enhanced Metafile Header Record	144
<b>EMF::EXTPEN</b>	
Extended Graphics Pen	146
<b>EMF::FONT</b>	
Graphics Font	147
<b>EMF::GLOBALOBJECTS</b>	
	149
<b>EMF::GRAPHICSOBJECT</b>	
A global graphics object	154
<b>EMF::INTARRAY</b>	
Represent an array of integers in a simple way	155
<b>EMF::METAFILEDEVICECONTEXT</b>	
Graphics Device Context	156
<b>EMF::METARECORD</b>	
The base class of all metafile records	161
<b>EMF::OBJECT</b>	
Global GDI object	163
<b>EMF::PADDING</b>	
All metafile records must be padded out to a multiple of 4 bytes	165
<b>EMF::PALETTE</b>	
Graphics Palette	166
<b>EMF::PEN</b>	
Graphics Pen	167
<b>EMF::POINT16ARRAY</b>	
Represent an array of 16-bit point in a simple way	169

**EMF::POINTLARRAY**

Represent an array of points in a simple way

170

**EMF::WCHARSTR**

Represent a wide (UNICODE) character string in a simple way

171

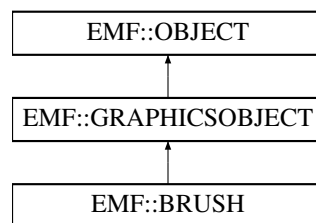
## 3 Data Structure Documentation

### 3.1 EMF::BRUSH Class Reference

Graphics Brush.

```
#include <libemf.h>
```

Inheritance diagram for EMF::BRUSH:



#### Public Member Functions

- [BRUSH](#) (const LOGBRUSH \*lbrush)
- OBJECTTYPE [getType](#) (void) const
- [METARECORD](#) \* [newEMR](#) (HDC dc, HGDIOBJ emf\_handle)

#### Additional Inherited Members

#### 3.1.1 Detailed Description

Graphics Brush.

Brushes are used for filling shapes.

#### 3.1.2 Constructor & Destructor Documentation

**3.1.2.1 BRUSH()** `EMF::BRUSH::BRUSH (`  
`const LOGBRUSH * lbrush ) [inline]`

## Parameters

<i>lbrush</i>	the (logical?) brush definition.
---------------	----------------------------------

## 3.1.3 Member Function Documentation

**3.1.3.1 getType()** OBJECTTYPE EMF::BRUSH::getType (   
 void ) const [inline], [virtual]

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.1.3.2 newEMR()** METARECORD\* EMF::BRUSH::newEMR (   
 HDC *dc*,   
 HGDIOBJ *emf\_handle* ) [inline], [virtual]

Return a new metarecord for this object. And record its selection into the given device context.

## Parameters

<i>dc</i>	handle of device context into which this object is being selected.
<i>emf_handle</i>	the EMF handle associated with the <a href="#">BRUSH</a> .

Implements [EMF::GRAPHICSOBJECT](#).

References EMF::GRAPHICSOBJECT::contexts.

The documentation for this class was generated from the following file:

- libemf.h

## 3.2 EMF::BYTEARRAY Struct Reference

Represent a byte array in a simple way.

```
#include <libemf.h>
```

## Public Member Functions

- [BYTEARRAY](#) (BYTE \*const array, const int n)



## Data Fields

- `BYTE *const array_`  
*Array of unsigned bytes.*
- `const int n_`  
*Number of bytes in array.*

### 3.2.1 Detailed Description

Represent a byte array in a simple way.

Evidently, an unsigned array of bytes with no particular encoding implied.

### 3.2.2 Constructor & Destructor Documentation

**3.2.2.1 BYTEARRAY()** `EMF::BYTEARRAY::BYTEARRAY (`  
 `BYTE *const array,`  
 `const int n ) [inline]`

Simple constructor.

#### Parameters

<i>array</i>	pointer to array of bytes
<i>n</i>	number of bytes in array

The documentation for this struct was generated from the following file:

- `libemf.h`

## 3.3 EMF::CHARSTR Struct Reference

Represent an ASCII character string in a simple way.

```
#include <libemf.h>
```

### Public Member Functions

- `CHARSTR` (`CHAR *const string`, `const int length`)

## Data Fields

- `CHAR *const string_`  
*Array of single byte characters.*
- `const int length_`  
*Number of single byte characters in array.*

### 3.3.1 Detailed Description

Represent an ASCII character string in a simple way.

ASCII strings don't have to be byte swapped, but this structure allows us to provide a uniform stream-like interface for writing out all the components of metafiles.

### 3.3.2 Constructor & Destructor Documentation

**3.3.2.1 CHARSTR()** `EMF::CHARSTR::CHARSTR (`  
`CHAR *const string,`  
`const int length ) [inline]`

Simple constructor.

#### Parameters

<i>string</i>	pointer to array of single byte characters.
<i>length</i>	number of bytes in array.

The documentation for this struct was generated from the following file:

- libemf.h

## 3.4 EMF::DATASTREAM Class Reference

Support different endian modes when reading and writing the metafile.

```
#include <libemf.h>
```

### Public Member Functions

- [DATASTREAM](#) (::FILE \*fp=0)
- void [setStream](#) (::FILE \*fp)
- [DATASTREAM](#) & [operator<<](#) (const BYTE &byte)
- [DATASTREAM](#) & [operator>>](#) (BYTE &byte)
- [DATASTREAM](#) & [operator<<](#) (const WORD &word)
- [DATASTREAM](#) & [operator>>](#) (WORD &word)
- [DATASTREAM](#) & [operator<<](#) (const INT16 &word)
- [DATASTREAM](#) & [operator>>](#) (INT16 &word)
- [DATASTREAM](#) & [operator<<](#) (const DWORD &dword)
- [DATASTREAM](#) & [operator>>](#) (DWORD &dword)
- [DATASTREAM](#) & [operator<<](#) (const LONG &long\_)
- [DATASTREAM](#) & [operator>>](#) (LONG &long\_)
- [DATASTREAM](#) & [operator<<](#) (const INT &int\_)
- [DATASTREAM](#) & [operator>>](#) (INT &int\_)

- `DATASTREAM & operator<<` (const `UINT` &uint)
- `DATASTREAM & operator>>` (`UINT` &uint)
- `DATASTREAM & operator<<` (const `FLOAT` &float\_)
- `DATASTREAM & operator>>` (`FLOAT` &float\_)
- `DATASTREAM & operator<<` (const `PADDING` &padding)
- `DATASTREAM & operator<<` (const `RECTL` &rectl)
- `DATASTREAM & operator>>` (`RECTL` &rectl)
- `DATASTREAM & operator<<` (const `SIZEL` &szel)
- `DATASTREAM & operator>>` (`SIZEL` &szel)
- `DATASTREAM & operator<<` (const `WCHARSTR` &wcharstr)
- `DATASTREAM & operator>>` (`WCHARSTR` &wcharstr)
- `DATASTREAM & operator<<` (const `CHARSTR` &charstr)
- `DATASTREAM & operator>>` (`CHARSTR` &charstr)
- `DATASTREAM & operator<<` (const `::EMR` &emr)
- `DATASTREAM & operator>>` (`::EMR` &emr)
- `DATASTREAM & operator<<` (const `POINT` &point)
- `DATASTREAM & operator>>` (`POINT` &point)
- `DATASTREAM & operator<<` (const `POINTL` &pointl)
- `DATASTREAM & operator>>` (`POINTL` &pointl)
- `DATASTREAM & operator<<` (const `POINT16` &point)
- `DATASTREAM & operator>>` (`POINT16` &point)
- `DATASTREAM & operator<<` (const `XFORM` &xform)
- `DATASTREAM & operator>>` (`XFORM` &xform)
- `DATASTREAM & operator<<` (const `BYTEARRAY` &array)
- `DATASTREAM & operator>>` (`BYTEARRAY` &array)
- `DATASTREAM & operator<<` (const `POINTLARRAY` &array)
- `DATASTREAM & operator>>` (`POINTLARRAY` &array)
- `DATASTREAM & operator<<` (const `POINT16ARRAY` &array)
- `DATASTREAM & operator>>` (`POINT16ARRAY` &array)
- `DATASTREAM & operator<<` (const `INTARRAY` &array)
- `DATASTREAM & operator>>` (`INTARRAY` &array)
- `DATASTREAM & operator<<` (const `DWORDARRAY` &array)
- `DATASTREAM & operator>>` (`DWORDARRAY` &array)
- `DATASTREAM & operator<<` (const `::EMRTEXT` &text)
- `DATASTREAM & operator>>` (`::EMRTEXT` &text)
- `DATASTREAM & operator<<` (const `LOGPEN` &pen)
- `DATASTREAM & operator>>` (`LOGPEN` &pen)
- `DATASTREAM & operator<<` (const `EXTLOGPEN` &pen)
- `DATASTREAM & operator>>` (`EXTLOGPEN` &pen)
- `DATASTREAM & operator<<` (const `LOGBRUSH` &brush)
- `DATASTREAM & operator>>` (`LOGBRUSH` &brush)
- `DATASTREAM & operator<<` (const `LOGFONTW` &font)
- `DATASTREAM & operator>>` (`LOGFONTW` &font)
- `DATASTREAM & operator<<` (const `PANOSE` &panose)
- `DATASTREAM & operator>>` (`PANOSE` &panose)
- `DATASTREAM & operator<<` (const `EXTLOGFONTW` &font)
- `DATASTREAM & operator>>` (`EXTLOGFONTW` &font)
- `DATASTREAM & operator<<` (const `LOGPALETTE` &palette)
- `DATASTREAM & operator>>` (`LOGPALETTE` &palette)

### 3.4.1 Detailed Description

Support different endian modes when reading and writing the metafile.

To support different endian modes, rather than just writing the structures directly to a file via `fwrite( &emr, ...)`, we have to write each element of the structure separately, swapping bytes as necessary. `datastream` supports this. Remarkably similar to the `QDataStream` class from Qt. So, too, for reading.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 DATASTREAM() `EMF::DATASTREAM::DATASTREAM (` `::FILE * fp = 0 ) [inline]`

Constructor for [DATASTREAM](#).

##### Parameters

<i>fp</i>	optional file pointer (but must be assigned before any output occurs.)
-----------	--

### 3.4.3 Member Function Documentation

#### 3.4.3.1 operator<<() [1/31] [DATASTREAM&](#) `EMF::DATASTREAM::operator<< (` `const ::EMR & emr ) [inline]`

Output an Enhanced Metafile Record header.

##### Parameters

<i>emr</i>	Enhanced Metafile Record header to output.
------------	--

#### 3.4.3.2 operator<<() [2/31] [DATASTREAM&](#) `EMF::DATASTREAM::operator<< (` `const ::EMRTEXT & text ) [inline]`

Output an Enhanced Metafile Text Record.

##### Parameters

<i>text</i>	Enhanced Metafile Text Record to output.
-------------	--

#### 3.4.3.3 operator<<() [3/31] [DATASTREAM&](#) `EMF::DATASTREAM::operator<< (` `const BYTE & byte ) [inline]`

Output a byte to the stream (not swabbed or anything).

##### Parameters

<i>byte</i>	byte to output.
-------------	-----------------

**3.4.3.4 operator<<()** [4/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [BYTEARRAY](#) & *array* ) [inline]

Output an array of BYTES.

Parameters

<i>array</i>	array of BYTES to output.
--------------	---------------------------

References EMF::BYTEARRAY::array\_, and EMF::BYTEARRAY::n\_.

**3.4.3.5 operator<<()** [5/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [CHARSTR](#) & *charstr* ) [inline]

Output a single byte character string.

Parameters

<i>charstr</i>	structure to output.
----------------	----------------------

References EMF::CHARSTR::length\_, and EMF::CHARSTR::string\_.

**3.4.3.6 operator<<()** [6/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [DWORD](#) & *dword* ) [inline]

Output a double word (long) to the stream (swabbed).

Parameters

<i>dword</i>	double word (long) to output.
--------------	-------------------------------

**3.4.3.7 operator<<()** [7/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [DWORDARRAY](#) & *array* ) [inline]

Output an array of double words (longs).

Parameters

<i>array</i>	array of double words (longs) to output.
--------------	--

References EMF::DWORDARRAY::dwords\_, and EMF::DWORDARRAY::n\_.

**3.4.3.8 operator<<()** [8/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
 const EXTLOGFONTW & *font* ) [inline]

Output an Extended Logical Font definition (using WCHAR strings).

#### Parameters

<i>font</i>	Extended Logical Font definition to output.
-------------	---

**3.4.3.9 operator<<()** [9/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
 const EXTLOGPEN & *pen* ) [inline]

Output an Extended Logical Pen definition.

#### Parameters

<i>pen</i>	Extended Logical Pen definition to output.
------------	--

**3.4.3.10 operator<<()** [10/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
 const FLOAT & *float\_* ) [inline]

Output a single precision float to the stream (swabbed).

#### Parameters

<i>float</i> ↔	single precision float to output.
—	

**3.4.3.11 operator<<()** [11/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
 const INT & *int\_* ) [inline]

Output a (long) int to the stream (swabbed).

#### Parameters

<i>int</i> ↔	(long) int to output.
—	

**3.4.3.12 operator<<()** [12/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const INT16 & *word* ) [inline]

Output a (short, 16-bit) word to the stream (swabbed).

**Parameters**

<i>word</i>	(short, 16-bit) word to output.
-------------	---------------------------------

**3.4.3.13 operator<<()** [13/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const INTARRAY & *array* ) [inline]

Output an array of (long) ints.

**Parameters**

<i>array</i>	array of (long) ints to output.
--------------	---------------------------------

References EMF::INTARRAY::ints\_, and EMF::INTARRAY::n\_.

**3.4.3.14 operator<<()** [14/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const LOGBRUSH & *brush* ) [inline]

Output a Logical Brush definition.

**Parameters**

<i>brush</i>	Logical Brush definition to output.
--------------	-------------------------------------

**3.4.3.15 operator<<()** [15/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const LOGFONTW & *font* ) [inline]

Output a Logical Font definition (using WCHAR strings).

**Parameters**

<i>font</i>	Logical Font definition to output.
-------------	------------------------------------

**3.4.3.16 operator<<()** [16/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const LOGPALETTE & *palette* ) [inline]

Output a Logical Palette.

Parameters

<i>palette</i>	Logical Palette to output.
----------------	----------------------------

**3.4.3.17 operator<<()** [17/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const LOGPEN & *pen* ) [inline]

Output a Logical Pen definition.

Parameters

<i>pen</i>	Logical Pen definition to output.
------------	-----------------------------------

**3.4.3.18 operator<<()** [18/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const LONG & *long\_* ) [inline]

Output a long int to the stream (swabbed).

Parameters

<i>long</i> ↔ —	long int to output.
--------------------	---------------------

**3.4.3.19 operator<<()** [19/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [PADDING](#) & *padding* ) [inline]

Output a series of '\0's to pad out a record.

Parameters

<i>padding</i>	simple padding structure (length and number of nulls).
----------------	--

References EMF::PADDING::padding\_, and EMF::PADDING::size\_.



**3.4.3.20 operator<<()** [20/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const PANOSE & *panose* ) [inline]

Output a Panose structure.

Parameters

<i>panose</i>	Panose structure to output.
---------------	-----------------------------

**3.4.3.21 operator<<()** [21/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const POINT & *point* ) [inline]

Output a POINT structure.

Parameters

<i>point</i>	POINT to output.
--------------	------------------

**3.4.3.22 operator<<()** [22/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const POINT16 & *point* ) [inline]

Output a POINT16 structure.

Parameters

<i>point</i>	POINT16 to output.
--------------	--------------------

**3.4.3.23 operator<<()** [23/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const [POINT16ARRAY](#) & *array* ) [inline]

Output an array of POINT16s.

Parameters

<i>array</i>	array of POINT16s to output.
--------------	------------------------------

References EMF::POINT16ARRAY::n\_, and EMF::POINT16ARRAY::points\_.

**3.4.3.24 operator<<()** [24/31] [DATASTREAM](#)& EMF::DATASTREAM::operator<< (   
const POINTL & *pointl* ) [inline]

Output a POINTL structure.

**Parameters**

<i>pointl</i>	POINTL to output.
---------------	-------------------

**3.4.3.25 operator<<()** [25/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [POINTLARRAY](#) & *array* ) [inline]

Output an array of POINTLs.

**Parameters**

<i>array</i>	array of POINTLs to output.
--------------	-----------------------------

References EMF::POINTLARRAY::n\_, and EMF::POINTLARRAY::points\_.

**3.4.3.26 operator<<()** [26/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const RECTL & *rectl* ) [inline]

Output a RECTL structure.

**Parameters**

<i>rectl</i>	structure to output.
--------------	----------------------

**3.4.3.27 operator<<()** [27/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const SIZEL & *szel* ) [inline]

Output a SIZEL structure.

**Parameters**

<i>szel</i>	structure to output.
-------------	----------------------

**3.4.3.28 operator<<()** [28/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const UINT & *uint* ) [inline]

Output a (long) unsigned int to the stream (swabbed).

## Parameters

<i>uint</i>	(long) unsigned int to output.
-------------	--------------------------------

**3.4.3.29 operator<<()** [29/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const [WCHARSTR](#) & *wcharstr* ) [inline]

Output a WCHAR string (note: the individual characters are swabbed).

## Parameters

<i>wcharstr</i>	structure to output.
-----------------	----------------------

References EMF::WCHARSTR::length\_, and EMF::WCHARSTR::string\_.

**3.4.3.30 operator<<()** [30/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const WORD & *word* ) [inline]

Output a (short) word to the stream (swabbed).

## Parameters

<i>word</i>	(short) word to output.
-------------	-------------------------

**3.4.3.31 operator<<()** [31/31] [DATASTREAM&](#) EMF::DATASTREAM::operator<< (   
const XFORM & *xform* ) [inline]

Output an XFORM structure.

## Parameters

<i>xform</i>	XFORM to output.
--------------	------------------

**3.4.3.32 operator>>()** [1/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> (   
::EMR & *emr* ) [inline]

Input an Enhanced Metafile Record header.

**Parameters**

<i>emr</i>	destination of Enhanced Metafile Record header.
------------	---

**3.4.3.33 operator>>()** [2/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 ::EMRTEXT & *text* ) [inline]

Input an Enhanced Metafile Text Record.

**Parameters**

<i>text</i>	destination of Enhanced Metafile Text Record.
-------------	---

**3.4.3.34 operator>>()** [3/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 BYTE & *byte* ) [inline]

Input a byte from the stream (not swabbed or anything).

**Parameters**

<i>byte</i>	destination for input byte.
-------------	-----------------------------

**3.4.3.35 operator>>()** [4/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 [BYTEARRAY](#) & *array* ) [inline]

Input an array of BYTES.

**Parameters**

<i>array</i>	destination of array of input BYTES.
--------------	--------------------------------------

References EMF::BYTEARRAY::array\_, and EMF::BYTEARRAY::n\_.

**3.4.3.36 operator>>()** [5/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 [CHARSTR](#) & *charstr* ) [inline]

Input a single byte character string.

## Parameters

<i>charstr</i>	destination of input CHAR string.
----------------	-----------------------------------

References EMF::CHARSTR::length\_, and EMF::CHARSTR::string\_.

**3.4.3.37 operator>>()** [6/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 DWORD & *dword* ) [inline]

Input a double word (long) from the stream (swabbed).

## Parameters

<i>dword</i>	destination for double word (long).
--------------	-------------------------------------

**3.4.3.38 operator>>()** [7/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 DWORDARRAY & *array* ) [inline]

Input an array of double words (longs).

## Parameters

<i>array</i>	destination of array of input double words (longs).
--------------	---

References EMF::DWORDARRAY::dwords\_, and EMF::DWORDARRAY::n\_.

**3.4.3.39 operator>>()** [8/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 EXTLOGFONTW & *font* ) [inline]

Input an Extended Logical Font definition (using WCHAR strings).

## Parameters

<i>font</i>	destination of Extended Logical Font definition.
-------------	--

**3.4.3.40 operator>>()** [9/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
 EXTLOGPEN & *pen* ) [inline]

Input an Extended Logical Pen definition.

**Parameters**

<i>pen</i>	destination of Extended Logical Pen definition.
------------	---

**3.4.3.41 operator>>()** [10/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
FLOAT & *float\_* ) [inline]

Input a single precision float from the stream (swabbed).

**Parameters**

<i>float</i> <sub>↔</sub>	destination for single precision float.
—	

**3.4.3.42 operator>>()** [11/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
INT & *int\_* ) [inline]

Input a (long) int from the stream (swabbed).

**Parameters**

<i>int</i> <sub>↔</sub>	destination for (long) int.
—	

**3.4.3.43 operator>>()** [12/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
INT16 & *word* ) [inline]

Input a (short, 16-bit) word from the stream (swabbed).

**Parameters**

<i>word</i>	destination for (short, 16-bit) word.
-------------	---------------------------------------

**3.4.3.44 operator>>()** [13/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
[INTARRAY](#) & *array* ) [inline]

Input an array of (long) ints.

## Parameters

<i>array</i>	destination of array of input (long) ints.
--------------	--

References EMF::INTARRAY::ints\_, and EMF::INTARRAY::n\_.

**3.4.3.45 operator>>()** [14/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( LOGBRUSH & *brush* ) [inline]

Input a Logical Brush definition.

## Parameters

<i>brush</i>	destination of Logical Brush definition.
--------------	--

**3.4.3.46 operator>>()** [15/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( LOGFONTW & *font* ) [inline]

Input a Logical Font definition (using WCHAR strings).

## Parameters

<i>font</i>	destination of Logical Font definition.
-------------	---

**3.4.3.47 operator>>()** [16/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( LOGPALETTE & *palette* ) [inline]

Input a Logical Palette.

## Parameters

<i>palette</i>	destination of input Logical Palette.
----------------	---------------------------------------

**3.4.3.48 operator>>()** [17/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( LOGPEN & *pen* ) [inline]

Input a Logical Pen definition.



**Parameters**

<i>pen</i>	destination of Logical Pen definition.
------------	--

**3.4.3.49 operator>>()** [18/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
LONG & *long\_* ) [inline]

Input a long int from the stream (swabbed).

**Parameters**

<i>long</i> $\leftrightarrow$	destination for long int.
—	

**3.4.3.50 operator>>()** [19/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
PANOSE & *panose* ) [inline]

Input a Panose structure.

**Parameters**

<i>panose</i>	destinatino of input Panose structure.
---------------	--

**3.4.3.51 operator>>()** [20/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
POINT & *point* ) [inline]

Input a POINT structure.

**Parameters**

<i>point</i>	destination of input POINT.
--------------	-----------------------------

**3.4.3.52 operator>>()** [21/30] [DATASTREAM](#)& EMF::DATASTREAM::operator>> (   
POINT16 & *point* ) [inline]

Input a POINT16 structure.

**Parameters**

<i>point</i>	destination of input POINT16.
--------------	-------------------------------

**3.4.3.53 operator>>()** [22/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( [POINT16ARRAY](#) & *array* ) [inline]

Input an array of POINT16s.

Parameters

<i>array</i>	destination of array of input POINT16s.
--------------	---

References EMF::POINT16ARRAY::n\_, and EMF::POINT16ARRAY::points\_.

**3.4.3.54 operator>>()** [23/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( [POINTL](#) & *pointl* ) [inline]

Input a POINTL structure.

Parameters

<i>pointl</i>	destination of input POINTL.
---------------	------------------------------

**3.4.3.55 operator>>()** [24/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( [POINTLARRAY](#) & *array* ) [inline]

Input an array of POINTLs.

Parameters

<i>array</i>	destination of array of input POINTLs.
--------------	--

References EMF::POINTLARRAY::n\_, and EMF::POINTLARRAY::points\_.

**3.4.3.56 operator>>()** [25/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( [RECTL](#) & *rectl* ) [inline]

Input a RECTL structure.

Parameters

<i>rectl</i>	destination of input RECTL.
--------------	-----------------------------

**3.4.3.57 operator>>()** [26/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> (   
 [SIZEL](#) & *szel* ) [inline]

Input a SIZEL structure.

**Parameters**

<i>szel</i>	destination of input SIZEL.
-------------	-----------------------------

**3.4.3.58 operator>>()** [27/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> (   
 [UINT](#) & *uint* ) [inline]

Input a (long) unsigned int from the stream (swabbed).

**Parameters**

<i>uint</i>	destination for (long) unsigned int.
-------------	--------------------------------------

**3.4.3.59 operator>>()** [28/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> (   
 [WCHARSTR](#) & *wcharstr* ) [inline]

Input a WCHAR string (note: the individual characters are swabbed.)

**Parameters**

<i>wcharstr</i>	destination of input WCHAR string.
-----------------	------------------------------------

References EMF::WCHARSTR::length\_, and EMF::WCHARSTR::string\_.

**3.4.3.60 operator>>()** [29/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> (   
 [WORD](#) & *word* ) [inline]

Input a (short) word from the stream (swabbed).

**Parameters**

<i>word</i>	destination for (short) word.
-------------	-------------------------------

**3.4.3.61 operator>>()** [30/30] [DATASTREAM&](#) EMF::DATASTREAM::operator>> ( XFORM & *xform* ) [inline]

Input an XFORM structure.

#### Parameters

<i>xform</i>	destination of input XFORM.
--------------	-----------------------------

**3.4.3.62 setStream()** void EMF::DATASTREAM::setStream ( ::FILE \* *fp* ) [inline]

Use the given FILE stream as the input/output destination.

#### Parameters

<i>fp</i>	file point for i/o.
-----------	---------------------

The documentation for this class was generated from the following file:

- libemf.h

## 3.5 EMF::DWORDARRAY Struct Reference

Represent an array of double word integers in a simple way.

```
#include <libemf.h>
```

### Public Member Functions

- [DWORDARRAY](#) (DWORD \*const dwords, const DWORD n)

### Data Fields

- DWORD \*const [dwords\\_](#)  
*Array of double words.*
- const DWORD [n\\_](#)  
*Number of double words in array.*

### 3.5.1 Detailed Description

Represent an array of double word integers in a simple way.

Allow an array of DWORD's to be written out at once.

### 3.5.2 Constructor & Destructor Documentation

**3.5.2.1 DWORDARRAY()** `EMF::DWORDARRAY::DWORDARRAY (`  
`DWORD *const dwords,`  
`const DWORD n ) [inline]`

simple constructor.

#### Parameters

<i>dwords</i>	pointer to double words.
<i>n</i>	number double words in array.

The documentation for this struct was generated from the following file:

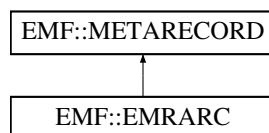
- libemf.h

## 3.6 EMF::EMRARC Class Reference

EMF Arc.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRARC:



#### Public Member Functions

- [EMRARC](#) (INT left, INT top, INT right, INT bottom, INT xstart, INT ystart, INT xend, INT yend)
- [EMRARC](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.6.1 Detailed Description

EMF Arc.

Draw an arc. Not sure what the specification here means, though.

### 3.6.2 Constructor & Destructor Documentation

**3.6.2.1 EMRARC()** [1/2] `EMF::EMRARC::EMRARC (`  
`INT left,`  
`INT top,`  
`INT right,`  
`INT bottom,`  
`INT xstart,`  
`INT ystart,`  
`INT xend,`  
`INT yend ) [inline]`

Take these descriptions with a grain of salt...

#### Parameters

<i>left</i>	x position of left edge of arc box.
<i>top</i>	y position of top edge of arc box.
<i>right</i>	x position of right edge of arc box.
<i>bottom</i>	y position bottom edge of arc box.
<i>xstart</i>	x position of arc start.
<i>ystart</i>	y position of arc start.
<i>xend</i>	x position of arc end.
<i>yend</i>	y position of arc end.

**3.6.2.2 EMRARC()** [2/2] `EMF::EMRARC::EMRARC (`  
`DATASTREAM & ds ) [inline]`

Construct an Arc record from the input datastream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.6.3 Member Function Documentation

**3.6.3.1 execute()** `void EMF::EMRARC::execute (`  
`METAFILEDEVICECONTEXT * source,`  
`HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.6.3.2 serialize()** `bool EMF::EMRARC::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.6.3.3 size()** `int EMF::EMRARC::size ( void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

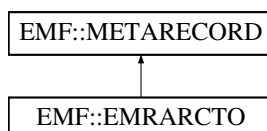
- libemf.h

## 3.7 EMF::EMRARCTO Class Reference

EMF Arc To.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRARCTO:



### Public Member Functions

- [EMRARCTO](#) (INT left, INT top, INT right, INT bottom, INT xstart, INT ystart, INT xend, INT yend)
- [EMRARCTO](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.7.1 Detailed Description

EMF Arc To.

Draw another arc. Not sure what the specification here means, though.

### 3.7.2 Constructor & Destructor Documentation

**3.7.2.1 EMRARCTO()** [1/2] `EMF::EMRARCTO::EMRARCTO (`  
`INT left,`  
`INT top,`  
`INT right,`  
`INT bottom,`  
`INT xstart,`  
`INT ystart,`  
`INT xend,`  
`INT yend ) [inline]`

Take these descriptions with a grain of salt...

#### Parameters

<i>left</i>	x position of left edge of arc box.
<i>top</i>	y position of top edge of arc box.
<i>right</i>	x position of right edge of arc box.
<i>bottom</i>	y position bottom edge of arc box.
<i>xstart</i>	x position of arc start.
<i>ystart</i>	y position of arc start.
<i>xend</i>	x position of arc end.
<i>yend</i>	y position of arc end.

**3.7.2.2 EMRARCTO()** [2/2] `EMF::EMRARCTO::EMRARCTO (`  
`DATASTREAM & ds ) [inline]`

Construct an ArcTo record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.7.3 Member Function Documentation



**3.7.3.1 execute()** `void EMF::EMRARCTO::execute (`  
    `METAFILEDVICECONTEXT * source,`  
    `HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.7.3.2 serialize()** `bool EMF::EMRARCTO::serialize (`  
    `DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.7.3.3 size()** `int EMF::EMRARCTO::size (`  
    `void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

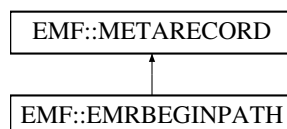
- libemf.h

## 3.8 EMF::EMRBEGINPATH Class Reference

EMF Begin Path.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRBEGINPATH:



## Public Member Functions

- [EMRBEGINPATH](#) (void)
- [EMRBEGINPATH](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.8.1 Detailed Description

EMF Begin Path.

Begin the current path definition.

### 3.8.2 Constructor & Destructor Documentation

**3.8.2.1 [EMRBEGINPATH\(\)](#) [1/2]** `EMF::EMRBEGINPATH::EMRBEGINPATH (void) [inline]`

Create a Begin Path record.

**3.8.2.2 [EMRBEGINPATH\(\)](#) [2/2]** `EMF::EMRBEGINPATH::EMRBEGINPATH (DATASTREAM & ds) [inline]`

Construct a BeginPath record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.8.3 Member Function Documentation

**3.8.3.1 [execute\(\)](#)** `void EMF::EMRBEGINPATH::execute (METAFILEDEVICECONTEXT * source, HDC dc) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.8.3.2 serialize()** `bool EMF::EMRBEGINPATH::serialize (   
     DATASTREAM ds )   [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.8.3.3 size()** `int EMF::EMRBEGINPATH::size (   
     void ) const   [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

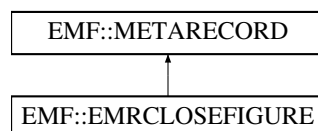
- libemf.h

## 3.9 EMF::EMRCLOSEFIGURE Class Reference

EMF Close Figure.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRCLOSEFIGURE:



### Public Member Functions

- [EMRCLOSEFIGURE](#) (void)
- [EMRCLOSEFIGURE](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.9.1 Detailed Description

EMF Close Figure.

Close the current figure.

### 3.9.2 Constructor & Destructor Documentation

**3.9.2.1 EMRCLOSEFIGURE() [1/2]** `EMF::EMRCLOSEFIGURE::EMRCLOSEFIGURE ( void ) [inline]`

Create a Close Figure record.

**3.9.2.2 EMRCLOSEFIGURE() [2/2]** `EMF::EMRCLOSEFIGURE::EMRCLOSEFIGURE ( DATASTREAM & ds ) [inline]`

Construct a CloseFigure record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.9.3 Member Function Documentation

**3.9.3.1 execute()** `void EMF::EMRCLOSEFIGURE::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.9.3.2 serialize()** `bool EMF::EMRCLOSEFIGURE::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.9.3.3 size()** `int EMF::EMRCLOSEFIGURE::size (void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

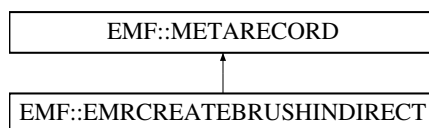
- libemf.h

## 3.10 EMF::EMRCREATEBRUSHINDIRECT Class Reference

EMF Brush.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRCREATEBRUSHINDIRECT:



### Public Member Functions

- [EMRCREATEBRUSHINDIRECT](#) ([BRUSH](#) \*brush, HGDIOBJ handle)
- [EMRCREATEBRUSHINDIRECT](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.10.1 Detailed Description

EMF Brush.

Create a new brush (used for filling shapes).

### 3.10.2 Constructor & Destructor Documentation

**3.10.2.1 EMRCREATEBRUSHINDIRECT()** [1/2] `EMRCREATEBRUSHINDIRECT::EMRCREATEBRUSHINDIRECT ( BRUSH * brush, HGDIOBJ handle )`

## Parameters

<i>brush</i>	an instance of a <a href="#">BRUSH</a> object.
<i>handle</i>	the <a href="#">BRUSH</a> object's handle.

### 3.10.2.2 EMRCREATEBRUSHINDIRECT() [2/2] `EMRCREATEBRUSHINDIRECT::EMRCREATEBRUSHINDIRECT ( DATASTREAM & ds )`

Create a CreateBrushIndirect record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.10.3 Member Function Documentation

### 3.10.3.1 execute() `void EMRCREATEBRUSHINDIRECT::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

### 3.10.3.2 serialize() `bool EMF::EMRCREATEBRUSHINDIRECT::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.10.3.3 size()** `int EMF::EMRCREATEBRUSHINDIRECT::size ( void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

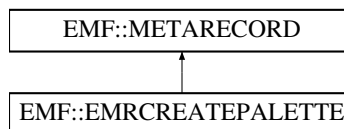
- [libemf.h](#)
- [libemf.cpp](#)

## 3.11 EMF::EMRCREATEPALETTE Class Reference

EMF Palette.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRCREATEPALETTE:



### Public Member Functions

- [EMRCREATEPALETTE](#) ([PALETTE](#) \*palette, HGDIOBJ handle)
- [EMRCREATEPALETTE](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

#### 3.11.1 Detailed Description

EMF Palette.

Create a new palette.

#### 3.11.2 Constructor & Destructor Documentation

**3.11.2.1 EMRCREATEPALETTE()** [1/2] `EMRCREATEPALETTE::EMRCREATEPALETTE ( PALETTE * palette, HGDIOBJ handle )`

## Parameters

<i>palette</i>	an instance of a <a href="#">PALETTE</a> object.
<i>handle</i>	the <a href="#">PALETTE</a> object's handle.

### 3.11.2.2 EMRCREATEPALETTE() [2/2]

```
EMF::EMRCREATEPALETTE::EMRCREATEPALETTE (
    DATASTREAM & ds )
```

Construct a CreatePalette record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.11.3 Member Function Documentation

### 3.11.3.1 execute()

```
void EMRCREATEPALETTE::execute (
    METAFILEDEVICECONTEXT * source,
    HDC dc ) const [virtual]
```

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

### 3.11.3.2 serialize()

```
bool EMF::EMRCREATEPALETTE::serialize (
    DATASTREAM ds ) [inline], [virtual]
```

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).



**3.11.3.3 size()** `int EMF::EMRCREATEPALETTE::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

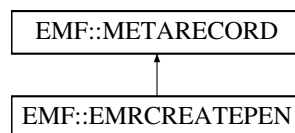
- `libemf.h`
- `libemf.cpp`

## 3.12 EMF::EMRCREATEPEN Class Reference

EMF Pen.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRCREATEPEN:



### Public Member Functions

- [EMRCREATEPEN](#) ([PEN](#) \*pen, HGDIOBJ handle)
- [EMRCREATEPEN](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.12.1 Detailed Description

EMF Pen.

Create a new pen (used for drawing lines, arcs, rectangles, etc.).

### 3.12.2 Constructor & Destructor Documentation

**3.12.2.1 EMRCREATEPEN()** [1/2] `EMRCREATEPEN::EMRCREATEPEN (`  
`PEN * pen,`  
`HGDIOBJ handle )`

## Parameters

<i>pen</i>	an instance of a <a href="#">PEN</a> object.
<i>handle</i>	the <a href="#">PEN</a> object's handle.

### 3.12.2.2 EMRCREATEPEN() [2/2] `EMRCREATEPEN::EMRCREATEPEN ( DATASTREAM & ds )`

Construct a CreatePen record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.12.3 Member Function Documentation

### 3.12.3.1 execute() `void EMRCREATEPEN::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

### 3.12.3.2 serialize() `bool EMF::EMRCREATEPEN::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

```
3.12.3.3 size() int EMF::EMRCREATEPEN::size (
    void ) const [inline], [virtual]
```

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

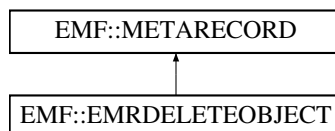
- libemf.h
- libemf.cpp

### 3.13 EMF::EMRDELETEOBJECT Class Reference

EMF Delete Object.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRDELETEOBJECT:



#### Public Member Functions

- [EMRDELETEOBJECT](#) (HGDIOBJ object)
- [EMRDELETEOBJECT](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.13.1 Detailed Description

EMF Delete Object.

Delete the given object, such as a pen, brush or font.

#### 3.13.2 Constructor & Destructor Documentation

```
3.13.2.1 EMRDELETEOBJECT() [1/2] EMF::EMRDELETEOBJECT::EMRDELETEOBJECT (
    HGDIOBJ object ) [inline]
```

## Parameters

<i>object</i>	the object to delete.
---------------	-----------------------

### 3.13.2.2 EMRDELETEOBJECT() [2/2] `EMF::EMRDELETEOBJECT::EMRDELETEOBJECT ( DATASTREAM & ds ) [inline]`

Construct a DeleteObject record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.13.3 Member Function Documentation

### 3.13.3.1 execute() `void EMRDELETEOBJECT::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

### 3.13.3.2 serialize() `bool EMF::EMRDELETEOBJECT::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.13.3.3 size()** `int EMF::EMRDELETEOBJECT::size ( void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

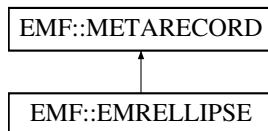
- libemf.h
- libemf.cpp

## 3.14 EMF::EMRELLIPSE Class Reference

EMF Ellipse.

`#include <libemf.h>`

Inheritance diagram for EMF::EMRELLIPSE:



### Public Member Functions

- [EMRELLIPSE](#) (INT left, INT top, INT right, INT bottom)
- [EMRELLIPSE](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.14.1 Detailed Description

EMF Ellipse.

Draw an ellipse. (I have no idea how the ellipse is defined!)

### 3.14.2 Constructor & Destructor Documentation

**3.14.2.1 EMRELLIPSE()** [1/2] `EMF::EMRELLIPSE::EMRELLIPSE ( INT left, INT top, INT right, INT bottom ) [inline]`

Take these descriptions with a grain of salt...

## Parameters

<i>left</i>	x position of left extrema of ellipse.
<i>top</i>	y position of top extrema of ellipse.
<i>right</i>	x position of right extrema of ellipse.
<i>bottom</i>	y position of bottom extrema of ellipse.

### 3.14.2.2 EMRELLIPSE() [2/2] `EMF::EMRELLIPSE::EMRELLIPSE ( DATASTREAM & ds ) [inline]`

Construct an Ellipse record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.14.3 Member Function Documentation

### 3.14.3.1 execute() `void EMF::EMRELLIPSE::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

### 3.14.3.2 serialize() `bool EMF::EMRELLIPSE::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.14.3.3 size()** `int EMF::EMRELLIPSE::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

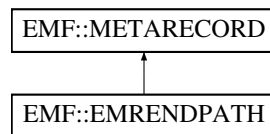
- libemf.h

## 3.15 EMF::EMRENDPATH Class Reference

EMF End Path.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRENDPATH:



### Public Member Functions

- [EMRENDPATH](#) (void)
- [EMRENDPATH](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.15.1 Detailed Description

EMF End Path.

End the current path definition.

### 3.15.2 Constructor & Destructor Documentation

**3.15.2.1 EMRENDPATH() [1/2]** `EMF::EMRENDPATH::EMRENDPATH (`  
`void ) [inline]`

Create an End Path record.

**3.15.2.2 EMRENDPATH() [2/2]** `EMF::EMRENDPATH::EMRENDPATH (`  
`DATASTREAM & ds ) [inline]`

Construct an EndPath record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.15.3 Member Function Documentation

**3.15.3.1 execute()** `void EMF::EMRENDPATH::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.15.3.2 serialize()** `bool EMF::EMRENDPATH::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.15.3.3 size()** `int EMF::EMRENDPATH::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

- [libemf.h](#)

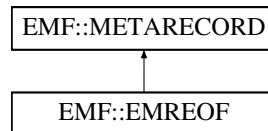


### 3.16 EMF::EMREOF Class Reference

EMF End of File Record.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMREOF:



#### Public Member Functions

- [EMREOF](#) (void)
- [EMREOF](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

#### 3.16.1 Detailed Description

EMF End of File Record.

Every metafile must have an End of File record. A palette may also be recorded in the EOF record, but it is currently unused by this library (all colors are specified in full RGB coordinates).

#### 3.16.2 Constructor & Destructor Documentation

**3.16.2.1 EMREOF()** [1/2] `EMF::EMREOF::EMREOF (void) [inline]`

Constructor contains no user serviceable parts.

**3.16.2.2 EMREOF()** [2/2] `EMF::EMREOF::EMREOF (DATASTREAM & ds) [inline]`

Construct an EOF record from the input stream

##### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.16.3 Member Function Documentation

**3.16.3.1 execute()** `void EMF::EMREOF::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.16.3.2 serialize()** `bool EMF::EMREOF::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.16.3.3 size()** `int EMF::EMREOF::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

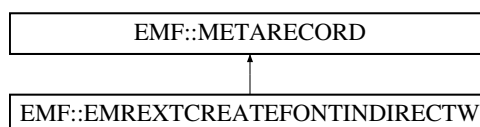
- libemf.h

## 3.17 EMF::EMREXTCREATEFONTINDIRECTW Class Reference

EMF Font.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMREXTCREATEFONTINDIRECTW:



## Public Member Functions

- [EMREXTCREATEFONTINDIRECTW](#) ([FONT](#) \*font, [HGDIOBJ](#) handle)
- [EMREXTCREATEFONTINDIRECTW](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, [HDC](#) dc) const

### 3.17.1 Detailed Description

EMF Font.

Create a new font.

### 3.17.2 Constructor & Destructor Documentation

**3.17.2.1 EMREXTCREATEFONTINDIRECTW() [1/2]** [EMREXTCREATEFONTINDIRECTW::EMREXTCREATEFONTI↔](#)  
 NDIRECTW (

```
    FONT * font,
    HGDIOBJ handle )
```

#### Parameters

<i>font</i>	an instance of a <a href="#">FONT</a> object.
<i>handle</i>	the <a href="#">FONT</a> object's handle.

**3.17.2.2 EMREXTCREATEFONTINDIRECTW() [2/2]** [EMREXTCREATEFONTINDIRECTW::EMREXTCREATEFONTI↔](#)  
 NDIRECTW (

```
    DATASTREAM & ds )
```

Construct a CreateFontIndirectW record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.17.3 Member Function Documentation

**3.17.3.1 execute()** void [EMREXTCREATEFONTINDIRECTW::execute](#) (   
[METAFILEDEVICECONTEXT](#) \* source,   
[HDC](#) dc ) const [virtual]

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

**3.17.3.2 serialize()** `bool EMF::EMREXTCREATEFONTINDIRECTW::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.17.3.3 size()** `int EMF::EMREXTCREATEFONTINDIRECTW::size (   
 void ) const [inline], [virtual]`

#### Returns

the size of the record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

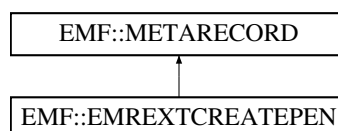
- libemf.h
- libemf.cpp

## 3.18 EMF::EMREXTCREATEPEN Class Reference

EMF Extended Pen.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMREXTCREATEPEN:



## Public Member Functions

- [EMREXTCREATEPEN](#) ([EXTPEN](#) \*pen, [HGDIOBJ](#) handle)
- [EMREXTCREATEPEN](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, [HDC](#) dc) const

### 3.18.1 Detailed Description

EMF Extended Pen.

Create a new pen (used for drawing lines, arcs, rectangles, etc.). Apparently uses extended attributes such as a bitmap mask.

### 3.18.2 Constructor & Destructor Documentation

**3.18.2.1 EMREXTCREATEPEN() [1/2]** `EMREXTCREATEPEN::EMREXTCREATEPEN (   
 EXTPEN * pen,   
 HGDIOBJ handle )`

#### Parameters

<i>pen</i>	an instance of a <a href="#">PEN</a> object.
<i>handle</i>	the <a href="#">PEN</a> object's handle.

**3.18.2.2 EMREXTCREATEPEN() [2/2]** `EMREXTCREATEPEN::EMREXTCREATEPEN (   
 DATASTREAM & ds )`

Construct a ExtCreatePen record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.18.3 Member Function Documentation

**3.18.3.1 execute()** `void EMREXTCREATEPEN::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

**3.18.3.2 `serialize()`** `bool EMF::EMREXTCREATEPEN::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.18.3.3 `size()`** `int EMF::EMREXTCREATEPEN::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

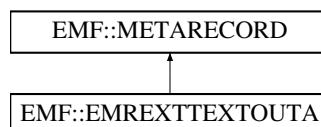
- `libemf.h`
- `libemf.cpp`

## 3.19 EMF::EMREXTTEXTOUTA Class Reference

EMF Extended Text Output ASCII.

```
#include <libemf.h>
```

Inheritance diagram for `EMF::EMREXTTEXTOUTA`:



## Public Member Functions

- [EMREXTTEXTOUTA](#) (const RECTL \*bounds, DWORD graphicsMode, FLOAT xScale, FLOAT yScale, const PEMRTEXT text, LPCSTR string, const INT \*dx)
- [EMREXTTEXTOUTA](#) (DATASTREAM &ds)
- [~EMREXTTEXTOUTA](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.19.1 Detailed Description

EMF Extended Text Output ASCII.

Draw this text string with the current font, in the color of the current pen and with the given text background color. Individual character positioning can be given in the dx array.

### 3.19.2 Constructor & Destructor Documentation

**3.19.2.1 EMREXTTEXTOUTA() [1/2]** `EMF::EMREXTTEXTOUTA::EMREXTTEXTOUTA (`  
`const RECTL * bounds,`  
`DWORD graphicsMode,`  
`FLOAT xScale,`  
`FLOAT yScale,`  
`const PEMRTEXT text,`  
`LPCSTR string,`  
`const INT * dx ) [inline]`

#### Parameters

<i>bounds</i>	bounding box of text string.
<i>graphicsMode</i>	(not entirely sure?)
<i>xScale</i>	width scale factor (of what?)
<i>yScale</i>	height scale factor (of what?)
<i>text</i>	a text metarecord containing the rendering style.
<i>string</i>	the text to render
<i>dx</i>	an array of positions for each character in string.

**3.19.2.2 EMREXTTEXTOUTA() [2/2]** `EMF::EMREXTTEXTOUTA::EMREXTTEXTOUTA (`  
`DATASTREAM & ds ) [inline]`

Construct a ExtTextOutA record from the input stream.



**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.19.2.3 ~EMREXTTEXTOUTA()** `EMF::EMREXTTEXTOUTA::~~EMREXTTEXTOUTA ( ) [inline]`

Destructor frees its copy of the string and its character offset array

**3.19.3 Member Function Documentation****3.19.3.1 execute()** `void EMF::EMREXTTEXTOUTA::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.19.3.2 serialize()** `bool EMF::EMREXTTEXTOUTA::serialize (   
 DATASTREAM ds ) [inline], [virtual]`**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.19.3.3 size()** `int EMF::EMREXTTEXTOUTA::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

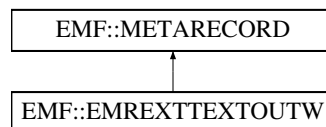
- libemf.h

## 3.20 EMF::EMREXTTEXTOUTW Class Reference

EMF Extended Text Output Wide character.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMREXTTEXTOUTW:



### Public Member Functions

- [EMREXTTEXTOUTW](#) (const RECTL \*bounds, DWORD graphicsMode, FLOAT xScale, FLOAT yScale, const PEMRTEXT text, LPCWSTR string, const INT \*dx)
- [EMREXTTEXTOUTW](#) (DATASTREAM &ds)
- [~EMREXTTEXTOUTW](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.20.1 Detailed Description

EMF Extended Text Output Wide character.

Draw this text string with the current font, in the color of the current pen and with the given text background color. Individual character positioning can be given in the dx array.

### 3.20.2 Constructor & Destructor Documentation

**3.20.2.1 EMREXTTEXTOUTW() [1/2]** `EMF::EMREXTTEXTOUTW::EMREXTTEXTOUTW (`  
`const RECTL * bounds,`  
`DWORD graphicsMode,`  
`FLOAT xScale,`  
`FLOAT yScale,`  
`const PEMRTEXT text,`  
`LPCWSTR string,`  
`const INT * dx ) [inline]`

#### Parameters

<i>bounds</i>	bounding box of text string.
<i>graphicsMode</i>	(not entirely sure?)
<i>xScale</i>	width scale factor (of what?)
<i>yScale</i>	height scale factor (of what?)
<i>text</i>	a text metarecord containing the rendering style.
<i>string</i>	the text to render
<i>dx</i>	an array of positions for each character in string.

**3.20.2.2 EMREXTTEXTOUTW()** [2/2] `EMF::EMREXTTEXTOUTW::EMREXTTEXTOUTW (   
 DATASTREAM & ds ) [inline]`

Construct a ExtTextOutA record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.20.2.3 ~EMREXTTEXTOUTW()** `EMF::EMREXTTEXTOUTW::~~EMREXTTEXTOUTW ( ) [inline]`

Destructor frees its copy of the string and its character offset array

### 3.20.3 Member Function Documentation

**3.20.3.1 execute()** `void EMF::EMREXTTEXTOUTW::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.20.3.2 serialize()** `bool EMF::EMREXTTEXTOUTW::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.20.3.3 size()** `int EMF::EMREXTTEXTOUTW::size (void) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

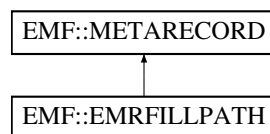
- libemf.h

## 3.21 EMF::EMRFILLPATH Class Reference

EMF Fill path.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRFILLPATH:



### Public Member Functions

- [EMRFILLPATH](#) (const RECTL \*bounds)
- [EMRFILLPATH](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.21.1 Detailed Description

EMF Fill path.

Fill the path.

### 3.21.2 Constructor & Destructor Documentation

**3.21.2.1 EMRFILLPATH()** [1/2] `EMF::EMRFILLPATH::EMRFILLPATH (const RECTL * bounds) [inline]`

**Parameters**

<i>bounds</i>	overall bounding box of polygon.
---------------	----------------------------------

**3.21.2.2 EMRFILLPATH() [2/2]** `EMF::EMRFILLPATH::EMRFILLPATH (   
 DATASTREAM & ds ) [inline]`

Create a FillPath record from input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.21.3 Member Function Documentation**

**3.21.3.1 execute()** `void EMF::EMRFILLPATH::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.21.3.2 serialize()** `bool EMF::EMRFILLPATH::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.21.3.3 size()** `int EMF::EMRFILLPATH::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

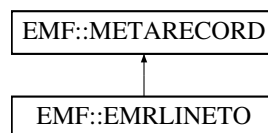
- `libemf.h`

## 3.22 EMF::EMRLINETO Class Reference

EMF Line To.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRLINETO:



### Public Member Functions

- [EMRLINETO](#) (INT x, INT y)
- [EMRLINETO](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.22.1 Detailed Description

EMF Line To.

Draw a line using the current pen to the given position.

### 3.22.2 Constructor & Destructor Documentation

**3.22.2.1 EMRLINETO()** [1/2] `EMF::EMRLINETO::EMRLINETO (`  
     INT x,  
     INT y )  `[inline]`

#### Parameters

<i>x</i>	x position to draw line to in logical coordinates.
<i>y</i>	y position to draw line to in logical coordinates.

**3.22.2.2 EMRLINETO()** [2/2] `EMF::EMRLINETO::EMRLINETO (`  
`DATASTREAM & ds ) [inline]`

Construct a LineTo record from the input stream.

Parameters

<i>ds</i>	Metafile datastream
-----------	---------------------

### 3.22.3 Member Function Documentation

**3.22.3.1 execute()** `void EMF::EMRLINETO::execute (`  
`METAFILEDEVICECONTEXT * source,`  
`HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.22.3.2 serialize()** `bool EMF::EMRLINETO::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream
-----------	---------------------

Implements [EMF::METARECORD](#).

**3.22.3.3 size()** `int EMF::EMRLINETO::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

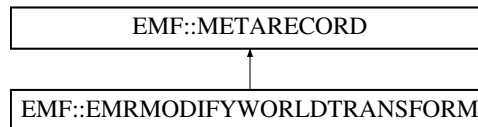
- libemf.h

### 3.23 EMF::EMRMODIFYWORLDTRANSFORM Class Reference

EMF Modify World Transform.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRMODIFYWORLDTRANSFORM:



#### Public Member Functions

- [EMRMODIFYWORLDTRANSFORM](#) (const XFORM \*transform, DWORD mode)
- [EMRMODIFYWORLDTRANSFORM](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.23.1 Detailed Description

EMF Modify World Transform.

Enhanced metafiles have a Coordinate Transformation which allows the contents to be rotated and transformed. Does not appear to work properly in StarOffice (but it's also possible I don't understand how it's supposed to work either).

#### 3.23.2 Constructor & Destructor Documentation

**3.23.2.1 EMRMODIFYWORLDTRANSFORM() [1/2]** `EMF::EMRMODIFYWORLDTRANSFORM::EMRMODIFYWORLDTRANSFORM ( const XFORM * transform, DWORD mode ) [inline]`

##### Parameters

<i>transform</i>	the transformation to apply
<i>mode</i>	the mode of the transformation application (namely, pre- or post-multiply)

**3.23.2.2 EMRMODIFYWORLDTRANSFORM() [2/2]** `EMF::EMRMODIFYWORLDTRANSFORM::EMRMODIFYWORLDTRANSFORM (`



```
DATASTREAM & ds ) [inline]
```

Construct a ModifyWorldTransform from the input datastream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.23.3 Member Function Documentation

**3.23.3.1 execute()** void EMF::EMRMODIFYWORLDTRANSFORM::execute (  
METAFILEDEVICECONTEXT \* source,  
HDC dc ) const [inline], [virtual]

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.23.3.2 serialize()** bool EMF::EMRMODIFYWORLDTRANSFORM::serialize (  
DATASTREAM ds ) [inline], [virtual]

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.23.3.3 size()** int EMF::EMRMODIFYWORLDTRANSFORM::size (  
void ) const [inline], [virtual]

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

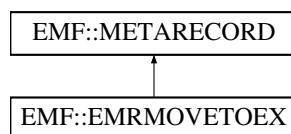
- libemf.h

## 3.24 EMF::EMRMOVETOEX Class Reference

EMF MoveTo (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRMOVETOEX:



### Public Member Functions

- [EMRMOVETOEX](#) (INT x, INT y)
- [EMRMOVETOEX](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.24.1 Detailed Description

EMF MoveTo (ex)

Move the drawing point to the given position.

### 3.24.2 Constructor & Destructor Documentation

**3.24.2.1 EMRMOVETOEX()** [1/2] `EMF::EMRMOVETOEX::EMRMOVETOEX (`  
     INT x,  
     INT y )  `[inline]`

Parameters

x	new x position in logical coordinates.
y	new y position in logical coordinates.

**3.24.2.2 EMRMOVETOEX()** [2/2] `EMF::EMRMOVETOEX::EMRMOVETOEX (`  
     DATASTREAM & ds )  `[inline]`

Construct a MoveToEx record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.24.3 Member Function Documentation**

**3.24.3.1 execute()** `void EMF::EMRMOVETOEX::execute (`  
    [METAFILEDEVICECONTEXT](#) \* *source*,  
    HDC *dc* ) const [inline], [virtual]

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.24.3.2 serialize()** `bool EMF::EMRMOVETOEX::serialize (`  
    [DATASTREAM](#) *ds* ) [inline], [virtual]

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.24.3.3 size()** `int EMF::EMRMOVETOEX::size (`  
    void ) const [inline], [virtual]

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

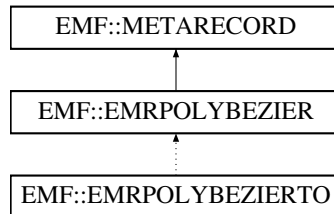
- libemf.h

### 3.25 EMF::EMRPOLYBEZIER Class Reference

EMF Polybezier.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYBEZIER:



#### Public Member Functions

- [EMRPOLYBEZIER](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYBEZIER](#) (DATASTREAM &ds)
- [~EMRPOLYBEZIER](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.25.1 Detailed Description

EMF Polybezier.

Draw a polygonal Bezier curve to (what?)

#### 3.25.2 Constructor & Destructor Documentation

**3.25.2.1 EMRPOLYBEZIER()** [1/2] `EMF::EMRPOLYBEZIER::EMRPOLYBEZIER (`  
     const RECTL \* *bounds*,  
     const POINT \* *points*,  
     INT *n* ) [inline]

##### Parameters

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

**3.25.2.2 EMRPOLYBEZIER()** [2/2] `EMF::EMRPOLYBEZIER::EMRPOLYBEZIER (   
 DATASTREAM & ds ) [inline]`

Construct a PolyBezier record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.25.2.3 ~EMRPOLYBEZIER()** `EMF::EMRPOLYBEZIER::~~EMRPOLYBEZIER ( ) [inline]`

Destructor frees a copy of the points it buffered.

### 3.25.3 Member Function Documentation

**3.25.3.1 execute()** `void EMF::EMRPOLYBEZIER::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO](#).

**3.25.3.2 serialize()** `bool EMF::EMRPOLYBEZIER::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO](#).

**3.25.3.3 size()** `int EMF::EMRPOLYBEZIER::size (void) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO](#).

The documentation for this class was generated from the following file:

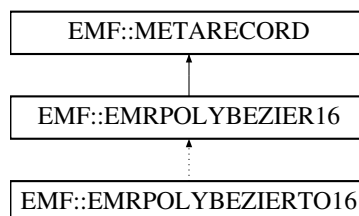
- libemf.h

## 3.26 EMF::EMRPOLYBEZIER16 Class Reference

EMF Polybezier16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYBEZIER16:



### Public Member Functions

- [EMRPOLYBEZIER16](#) (const RECTL \*bounds, const POINT16 \*points, INT n)
- [EMRPOLYBEZIER16](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYBEZIER16](#) (DATASTREAM &ds)
- [~EMRPOLYBEZIER16](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.26.1 Detailed Description

EMF Polybezier16.

Draw a polygonal Bezier curve to (what?) using 16-bit points.

### 3.26.2 Constructor & Destructor Documentation

**3.26.2.1 EMRPOLYBEZIER16()** [1/3] `EMF::EMRPOLYBEZIER16::EMRPOLYBEZIER16 (const RECTL * bounds, const POINT16 * points, INT n) [inline]`

**Parameters**

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

Referenced by EMRPOLYBEZIER16().

**3.26.2.2 EMRPOLYBEZIER16()** [2/3] `EMF::EMRPOLYBEZIER16::EMRPOLYBEZIER16 (`  
`const RECTL * bounds,`  
`const POINT * points,`  
`INT n ) [inline]`

Convenience constructor with POINTs.

**Parameters**

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

References EMRPOLYBEZIER16().

**3.26.2.3 EMRPOLYBEZIER16()** [3/3] `EMF::EMRPOLYBEZIER16::EMRPOLYBEZIER16 (`  
`DATASTREAM & ds ) [inline]`

Construct a PolyBezier record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.26.2.4 ~EMRPOLYBEZIER16()** `EMF::EMRPOLYBEZIER16::~~EMRPOLYBEZIER16 ( ) [inline]`

Destructor frees a copy of the points it buffered.

**3.26.3 Member Function Documentation**

**3.26.3.1 execute()** `void EMF::EMRPOLYBEZIER16::execute (`  
    `METAFILEDEVICECONTEXT * source,`  
    `HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.



## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO16](#).

**3.26.3.2 serialize()** `bool EMF::EMRPOLYBEZIER16::serialize ( DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO16](#).

**3.26.3.3 size()** `int EMF::EMRPOLYBEZIER16::size ( void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

Reimplemented in [EMF::EMRPOLYBEZIERTO16](#).

The documentation for this class was generated from the following file:

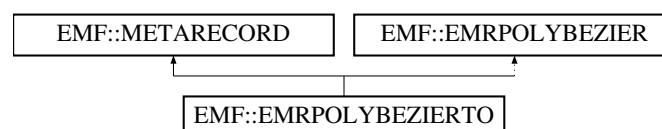
- libemf.h

## 3.27 EMF::EMRPOLYBEZIERTO Class Reference

EMF PolyBezierTo.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYBEZIERTO:



## Public Member Functions

- [EMRPOLYBEZIERTO](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYBEZIERTO](#) ([DATASTREAM](#) &ds)
- [~EMRPOLYBEZIERTO](#) ()
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.27.1 Detailed Description

EMF PolyBezierTo.

Draw a polygonal Bezier curve to (what?)

### 3.27.2 Constructor & Destructor Documentation

**3.27.2.1 [EMRPOLYBEZIERTO\(\)](#) [1/2]** `EMF::EMRPOLYBEZIERTO::EMRPOLYBEZIERTO ( const RECTL * bounds, const POINT * points, INT n ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

**3.27.2.2 [EMRPOLYBEZIERTO\(\)](#) [2/2]** `EMF::EMRPOLYBEZIERTO::EMRPOLYBEZIERTO ( DATASTREAM & ds ) [inline]`

Construct a PolyBezier record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.27.2.3 [~EMRPOLYBEZIERTO\(\)](#)** `EMF::EMRPOLYBEZIERTO::~~EMRPOLYBEZIERTO ( ) [inline]`

Destructor frees a copy of the points it buffered.

### 3.27.3 Member Function Documentation

**3.27.3.1 execute()** `void EMF::EMRPOLYBEZIERTO::execute (`  
`METAFILEDEVICECONTEXT * source,`  
`HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.27.3.2 serialize()** `bool EMF::EMRPOLYBEZIERTO::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.27.3.3 size()** `int EMF::EMRPOLYBEZIERTO::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

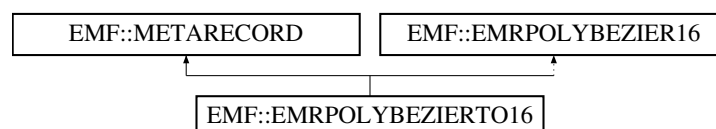
- libemf.h

## 3.28 EMF::EMRPOLYBEZIERTO16 Class Reference

EMF PolyBezierTo16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYBEZIERTO16:



## Public Member Functions

- [EMRPOLYBEZIERTO16](#) (const RECTL \*bounds, const POINT16 \*points, INT n)
- [EMRPOLYBEZIERTO16](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYBEZIERTO16](#) (DATASTREAM &ds)
- [~EMRPOLYBEZIERTO16](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.28.1 Detailed Description

EMF PolyBezierTo16.

Draw a polygonal Bezier curve to (what?) using 16-bit points

### 3.28.2 Constructor & Destructor Documentation

**3.28.2.1 EMRPOLYBEZIERTO16() [1/3]** `EMF::EMRPOLYBEZIERTO16::EMRPOLYBEZIERTO16 (`  
`const RECTL * bounds,`  
`const POINT16 * points,`  
`INT n ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

Referenced by [EMRPOLYBEZIERTO16\(\)](#).

**3.28.2.2 EMRPOLYBEZIERTO16() [2/3]** `EMF::EMRPOLYBEZIERTO16::EMRPOLYBEZIERTO16 (`  
`const RECTL * bounds,`  
`const POINT * points,`  
`INT n ) [inline]`

Convenience constructor with POINTs.

#### Parameters

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

References [EMRPOLYBEZIERTO16\(\)](#).

**3.28.2.3 [EMRPOLYBEZIERTO16\(\)](#)** [3/3] `EMF::EMRPOLYBEZIERTO16::EMRPOLYBEZIERTO16 (   
 DATASTREAM & ds ) [inline]`

Construct a PolyBezier record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.28.2.4 [~EMRPOLYBEZIERTO16\(\)](#)** `EMF::EMRPOLYBEZIERTO16::~~EMRPOLYBEZIERTO16 ( ) [inline]`

Destructor frees a copy of the points it buffered.

### 3.28.3 Member Function Documentation

**3.28.3.1 [execute\(\)](#)** `void EMF::EMRPOLYBEZIERTO16::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.28.3.2 [serialize\(\)](#)** `bool EMF::EMRPOLYBEZIERTO16::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.28.3.3 size()** `int EMF::EMRPOLYBEZIERTO16::size (void) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

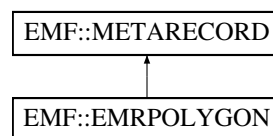
- libemf.h

## 3.29 EMF::EMRPOLYGON Class Reference

EMF Filled Polygon.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYGON:



### Public Member Functions

- [EMRPOLYGON](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYGON](#) (DATASTREAM &ds)
- [~EMRPOLYGON](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.29.1 Detailed Description

EMF Filled Polygon.

Draw a filled polygon.

### 3.29.2 Constructor & Destructor Documentation

**3.29.2.1 EMRPOLYGON()** [1/2] `EMF::EMRPOLYGON::EMRPOLYGON (const RECTL * bounds, const POINT * points, INT n) [inline]`

## Parameters

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>n</i>	number of vertices in points.

**3.29.2.2 EMRPOLYGON()** [2/2] `EMF::EMRPOLYGON::EMRPOLYGON (   
 DATASTREAM & ds ) [inline]`

Construct a Polygon record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.29.2.3 ~EMRPOLYGON()** `EMF::EMRPOLYGON::~~EMRPOLYGON ( ) [inline]`

Destructor frees a copy of the points it buffered.

### 3.29.3 Member Function Documentation

**3.29.3.1 execute()** `void EMF::EMRPOLYGON::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.29.3.2 serialize()** `bool EMF::EMRPOLYGON::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.29.3.3 size()** `int EMF::EMRPOLYGON::size (void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

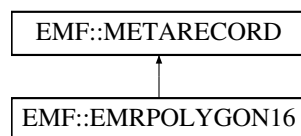
- libemf.h

### 3.30 EMF::EMRPOLYGON16 Class Reference

EMF Filled Polygon16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYGON16:



#### Public Member Functions

- [EMRPOLYGON16](#) (const RECTL \*bounds, const POINT \*points, INT16 n)
- [EMRPOLYGON16](#) (const RECTL \*bounds, const POINT16 \*points, INT16 n)
- [EMRPOLYGON16](#) (DATASTREAM &ds)
- [~EMRPOLYGON16](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.30.1 Detailed Description

EMF Filled Polygon16.

Draw a filled polygon (with 16-bit points).

#### 3.30.2 Constructor & Destructor Documentation

**3.30.2.1 EMRPOLYGON16()** [1/3] `EMF::EMRPOLYGON16::EMRPOLYGON16 (const RECTL * bounds, const POINT * points, INT16 n ) [inline]`



**Parameters**

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>n</i>	number of vertices in points.

Referenced by EMRPOLYGON16().

**3.30.2.2 EMRPOLYGON16()** [2/3] `EMF::EMRPOLYGON16::EMRPOLYGON16 (`  
`const RECTL * bounds,`  
`const POINT16 * points,`  
`INT16 n ) [inline]`

Additional constructor which takes a POINT16 array.

**Parameters**

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>n</i>	number of vertices in points.

References EMRPOLYGON16().

**3.30.2.3 EMRPOLYGON16()** [3/3] `EMF::EMRPOLYGON16::EMRPOLYGON16 (`  
`DATASTREAM & ds ) [inline]`

Construct a Polygon record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.30.2.4 ~EMRPOLYGON16()** `EMF::EMRPOLYGON16::~~EMRPOLYGON16 ( ) [inline]`

Destructor frees a copy of the points it buffered.

### 3.30.3 Member Function Documentation

**3.30.3.1 execute()** `void EMF::EMRPOLYGON16::execute (`  
    `METAFILEDEVICECONTEXT * source,`  
    `HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.30.3.2 serialize()** `bool EMF::EMRPOLYGON16::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.30.3.3 size()** `int EMF::EMRPOLYGON16::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

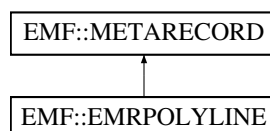
- libemf.h

## 3.31 EMF::EMRPOLYLINE Class Reference

EMF Polyline.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYLINE:



## Public Member Functions

- [EMRPOLYLINE](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [~EMRPOLYLINE](#) ()
- [EMRPOLYLINE](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.31.1 Detailed Description

EMF Polyline.

Draw a series of connected lines.

### 3.31.2 Constructor & Destructor Documentation

**3.31.2.1 EMRPOLYLINE()** [1/2] `EMF::EMRPOLYLINE::EMRPOLYLINE ( const RECTL * bounds, const POINT * points, INT n ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polyline.
<i>points</i>	array of polyline vertices.
<i>n</i>	number of vertices in points.

**3.31.2.2 ~EMRPOLYLINE()** `EMF::EMRPOLYLINE::~~EMRPOLYLINE ( ) [inline]`

Destructor frees a copy of the points it buffered.

**3.31.2.3 EMRPOLYLINE()** [2/2] `EMF::EMRPOLYLINE::EMRPOLYLINE ( DATASTREAM & ds ) [inline]`

Construct a Polyline record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.31.3 Member Function Documentation

**3.31.3.1 execute()** `void EMF::EMRPOLYLINE::execute (`  
`METAFILEDEVICECONTEXT * source,`  
`HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.31.3.2 serialize()** `bool EMF::EMRPOLYLINE::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.31.3.3 size()** `int EMF::EMRPOLYLINE::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

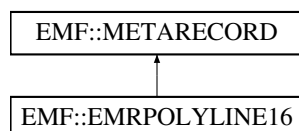
- libemf.h

## 3.32 EMF::EMRPOLYLINE16 Class Reference

EMF Polyline16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYLINE16:



## Public Member Functions

- [EMRPOLYLINE16](#) (const RECTL \*bounds, const POINT16 \*points, INT n)
- [EMRPOLYLINE16](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [~EMRPOLYLINE16](#) ()
- [EMRPOLYLINE16](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.32.1 Detailed Description

EMF Polyline16.

Draw a series of connected lines using 16-bit points.

### 3.32.2 Constructor & Destructor Documentation

**3.32.2.1 EMRPOLYLINE16() [1/3]** `EMF::EMRPOLYLINE16::EMRPOLYLINE16 (`  
`const RECTL * bounds,`  
`const POINT16 * points,`  
`INT n ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polyline.
<i>points</i>	array of polyline vertices.
<i>n</i>	number of vertices in points.

Referenced by [EMRPOLYLINE16\(\)](#).

**3.32.2.2 EMRPOLYLINE16() [2/3]** `EMF::EMRPOLYLINE16::EMRPOLYLINE16 (`  
`const RECTL * bounds,`  
`const POINT * points,`  
`INT n ) [inline]`

Constructor with POINTs.

#### Parameters

<i>bounds</i>	overall bounding box of polyline.
<i>points</i>	array of polyline vertices.
<i>n</i>	number of vertices in points.

References `EMRPOLYLINE16()`.

**3.32.2.3** `~EMRPOLYLINE16()` `EMF::EMRPOLYLINE16::~~EMRPOLYLINE16 ( ) [inline]`

Destructor frees a copy of the points it buffered.

**3.32.2.4** `EMRPOLYLINE16()` `[3/3] EMF::EMRPOLYLINE16::EMRPOLYLINE16 (   
 DATASTREAM & ds ) [inline]`

Construct a Polyline record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.32.3 Member Function Documentation

**3.32.3.1** `execute()` `void EMF::EMRPOLYLINE16::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.32.3.2** `serialize()` `bool EMF::EMRPOLYLINE16::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.32.3.3 size()** `int EMF::EMRPOLYLINE16::size (void) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

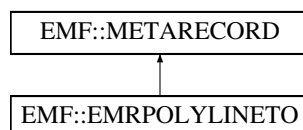
- libemf.h

### 3.33 EMF::EMRPOLYLINETO Class Reference

EMF PolylineTo.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYLINETO:



#### Public Member Functions

- [EMRPOLYLINETO](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYLINETO](#) (DATASTREAM &ds)
- [~EMRPOLYLINETO](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.33.1 Detailed Description

EMF PolylineTo.

Draw a polygonal line curve to (what?)

#### 3.33.2 Constructor & Destructor Documentation

**3.33.2.1 EMRPOLYLINETO()** [1/2] `EMF::EMRPOLYLINETO::EMRPOLYLINETO (const RECTL * bounds, const POINT * points, INT n) [inline]`



**Parameters**

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

**3.33.2.2 EMRPOLYLINETO()** [2/2] `EMF::EMRPOLYLINETO::EMRPOLYLINETO (   
 DATASTREAM & ds ) [inline]`

Construct a PolylineTo record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.33.2.3 ~EMRPOLYLINETO()** `EMF::EMRPOLYLINETO::~~EMRPOLYLINETO ( ) [inline]`

Destructor frees a copy of the points it buffered.

**3.33.3 Member Function Documentation**

**3.33.3.1 execute()** `void EMF::EMRPOLYLINETO::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.33.3.2 serialize()** `bool EMF::EMRPOLYLINETO::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.33.3.3 size()** `int EMF::EMRPOLYLINETO::size (void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

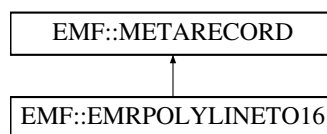
- libemf.h

## 3.34 EMF::EMRPOLYLINETO16 Class Reference

EMF PolylineTo16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYLINETO16:



### Public Member Functions

- [EMRPOLYLINETO16](#) (const RECTL \*bounds, const POINT16 \*points, INT n)
- [EMRPOLYLINETO16](#) (const RECTL \*bounds, const POINT \*points, INT n)
- [EMRPOLYLINETO16](#) (DATASTREAM &ds)
- [~EMRPOLYLINETO16](#) ()
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.34.1 Detailed Description

EMF PolylineTo16.

Draw a polygonal line curve to (what?)

### 3.34.2 Constructor & Destructor Documentation

**3.34.2.1 EMRPOLYLINETO16() [1/3]** `EMF::EMRPOLYLINETO16::EMRPOLYLINETO16 (const RECTL * bounds, const POINT16 * points, INT n ) [inline]`

**Parameters**

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

Referenced by EMRPOLYLINETO16().

**3.34.2.2 EMRPOLYLINETO16()** [2/3] `EMF::EMRPOLYLINETO16::EMRPOLYLINETO16 (`  
`const RECTL * bounds,`  
`const POINT * points,`  
`INT n ) [inline]`

Convenience constructor with POINTs.

**Parameters**

<i>bounds</i>	overall bounding box of polybezier curve.
<i>points</i>	array of polybezier vertices.
<i>n</i>	number of vertices in points.

References EMRPOLYLINETO16().

**3.34.2.3 EMRPOLYLINETO16()** [3/3] `EMF::EMRPOLYLINETO16::EMRPOLYLINETO16 (`  
`DATASTREAM & ds ) [inline]`

Construct a PolylineTo record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.34.2.4 ~EMRPOLYLINETO16()** `EMF::EMRPOLYLINETO16::~~EMRPOLYLINETO16 ( ) [inline]`

Destructor frees a copy of the points it buffered.

**3.34.3 Member Function Documentation**

**3.34.3.1 execute()** void EMF::EMRPOLYLINETO16::execute (  
    METAFILEDEVICECONTEXT \* *source*,  
    HDC *dc* ) const [inline], [virtual]

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.34.3.2 `serialize()`** `bool EMF::EMRPOLYLINETO16::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.34.3.3 `size()`** `int EMF::EMRPOLYLINETO16::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

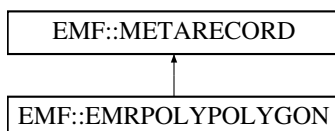
- libemf.h

## 3.35 EMF::EMRPOLYPOLYGON Class Reference

EMF Poly Polygon.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYPOLYGON:



## Public Member Functions

- [EMRPOLYPOLYGON](#) (const RECTL \*bounds, const POINT \*points, const INT \*counts, UINT polygons)
- [~EMRPOLYPOLYGON](#) ()
- [EMRPOLYPOLYGON](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.35.1 Detailed Description

EMF Poly Polygon.

Draw several filled polygons.

### 3.35.2 Constructor & Destructor Documentation

**3.35.2.1 EMRPOLYPOLYGON()** [1/2] `EMF::EMRPOLYPOLYGON::EMRPOLYPOLYGON ( const RECTL * bounds, const POINT * points, const INT * counts, UINT polygons ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>counts</i>	array of number of vertices in each polygon.
<i>polygons</i>	number of polygons.

**3.35.2.2 ~EMRPOLYPOLYGON()** `EMF::EMRPOLYPOLYGON::~~EMRPOLYPOLYGON ( ) [inline]`

Destructor frees a copy of the counts and points it buffered.

**3.35.2.3 EMRPOLYPOLYGON()** [2/2] `EMF::EMRPOLYPOLYGON::EMRPOLYPOLYGON ( DATASTREAM & ds ) [inline]`

Construct a Polygon record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.35.3 Member Function Documentation

**3.35.3.1 execute()** `void EMF::EMRPOLYPOLYGON::execute (`  
`METAFILEDEVICECONTEXT * source,`  
`HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.35.3.2 serialize()** `bool EMF::EMRPOLYPOLYGON::serialize (`  
`DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.35.3.3 size()** `int EMF::EMRPOLYPOLYGON::size (`  
`void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

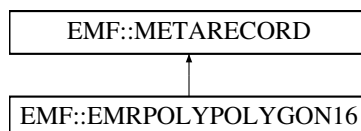
- libemf.h

## 3.36 EMF::EMRPOLYPOLYGON16 Class Reference

EMF Poly Polygon16.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRPOLYPOLYGON16:



## Public Member Functions

- [EMRPOLYPOLYGON16](#) (const RECTL \*bounds, const POINT \*points, const INT \*counts, UINT polygons)
- [EMRPOLYPOLYGON16](#) (const RECTL \*bounds, const POINT16 \*points, const INT \*counts, UINT16 polygons)
- [~EMRPOLYPOLYGON16](#) ()
- [EMRPOLYPOLYGON16](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.36.1 Detailed Description

EMF Poly Polygon16.

Draw several filled polygons (with 16-bit points).

### 3.36.2 Constructor & Destructor Documentation

**3.36.2.1 EMRPOLYPOLYGON16() [1/3]** `EMF::EMRPOLYPOLYGON16::EMRPOLYPOLYGON16 ( const RECTL * bounds, const POINT * points, const INT * counts, UINT polygons ) [inline]`

#### Parameters

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>counts</i>	array of number of vertices in each polygon.
<i>polygons</i>	number of polygons.

Referenced by [EMRPOLYPOLYGON16\(\)](#).

**3.36.2.2 EMRPOLYPOLYGON16() [2/3]** `EMF::EMRPOLYPOLYGON16::EMRPOLYPOLYGON16 ( const RECTL * bounds, const POINT16 * points, const INT * counts, UINT16 polygons ) [inline]`

Additional constructor which takes a POINT16 structure.

#### Parameters

<i>bounds</i>	overall bounding box of polygon.
<i>points</i>	array of polygon vertices.
<i>counts</i>	array of number of vertices in each polygon.
<i>polygons</i>	number of polygons.



References `EMRPOLYPOLYGON16()`.

**3.36.2.3** `~EMRPOLYPOLYGON16()` `EMF::EMRPOLYPOLYGON16::~~EMRPOLYPOLYGON16 ( ) [inline]`

Destructor frees a copy of the counts and points it buffered.

**3.36.2.4** `EMRPOLYPOLYGON16()` `[3/3] EMF::EMRPOLYPOLYGON16::EMRPOLYPOLYGON16 (   
 DATASTREAM & ds ) [inline]`

Construct a Polygon record from the input stream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.36.3 Member Function Documentation

**3.36.3.1** `execute()` `void EMF::EMRPOLYPOLYGON16::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.36.3.2** `serialize()` `bool EMF::EMRPOLYPOLYGON16::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.36.3.3 size()** `int EMF::EMRPOLYPOLYGON16::size (void) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

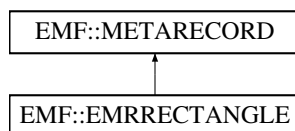
- libemf.h

## 3.37 EMF::EMRRECTANGLE Class Reference

EMF Rectangle.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRRECTANGLE:



### Public Member Functions

- [EMRRECTANGLE](#) (INT left, INT top, INT right, INT bottom)
- [EMRRECTANGLE](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.37.1 Detailed Description

EMF Rectangle.

Draw a rectangle.

### 3.37.2 Constructor & Destructor Documentation

**3.37.2.1 EMRRECTANGLE()** [1/2] `EMF::EMRRECTANGLE::EMRRECTANGLE (INT left, INT top, INT right, INT bottom) [inline]`

**Parameters**

<i>left</i>	x position of left side of rectangle.
<i>top</i>	y position of top side of rectangle.
<i>right</i>	x position of right edge of rectangle.
<i>bottom</i>	y position of bottom edge of rectangle.

**3.37.2.2 EMRRECTANGLE()** [2/2] `EMF::EMRRECTANGLE::EMRRECTANGLE (   
 DATASTREAM & ds ) [inline]`

Construct a Rectangle record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.37.3 Member Function Documentation****3.37.3.1 execute()** `void EMF::EMRRECTANGLE::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.37.3.2 serialize()** `bool EMF::EMRRECTANGLE::serialize (   
 DATASTREAM ds ) [inline], [virtual]`**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.37.3.3 size()** `int EMF::EMRRECTANGLE::size ( void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

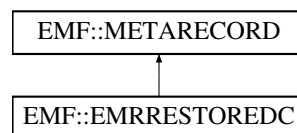
- libemf.h

## 3.38 EMF::EMRRESTOREDC Class Reference

EMF Restore DC.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRRESTOREDC:



### Public Member Functions

- [EMRRESTOREDC](#) (INT n)
- [EMRRESTOREDC](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.38.1 Detailed Description

EMF Restore DC.

Use the stored device context in this context(?)

### 3.38.2 Constructor & Destructor Documentation

**3.38.2.1 EMRRESTOREDC()** [1/2] `EMF::EMRRESTOREDC::EMRRESTOREDC ( INT n ) [inline]`

Create a Restore DC record.

**3.38.2.2 EMRRESTOREDC()** [2/2] `EMF::EMRRESTOREDC::EMRRESTOREDC ( DATASTREAM & ds ) [inline]`

Construct an Restoredc record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.38.3 Member Function Documentation**

**3.38.3.1 execute()** `void EMF::EMRRESTOREDC::execute (`  
    [METAFILEDEVICECONTEXT](#) \* *source*,  
    HDC *dc* ) const [inline], [virtual]

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.38.3.2 serialize()** `bool EMF::EMRRESTOREDC::serialize (`  
    [DATASTREAM](#) *ds* ) [inline], [virtual]

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.38.3.3 size()** `int EMF::EMRRESTOREDC::size (`  
    void ) const [inline], [virtual]

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

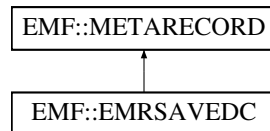
- libemf.h

### 3.39 EMF::EMRSAVEDC Class Reference

EMF Save DC.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSAVEDC:



#### Public Member Functions

- [EMRSAVEDC](#) (void)
- [EMRSAVEDC](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.39.1 Detailed Description

EMF Save DC.

Save the device context (i.e., push contents on a stack of some variety?)

#### 3.39.2 Constructor & Destructor Documentation

**3.39.2.1 EMRSAVEDC()** [1/2] EMF::EMRSAVEDC::EMRSAVEDC (void) [inline]

Create a Save DC record.

**3.39.2.2 EMRSAVEDC()** [2/2] EMF::EMRSAVEDC::EMRSAVEDC (DATASTREAM & ds) [inline]

Construct an Savedc record from the input stream.

##### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.39.3 Member Function Documentation

**3.39.3.1 execute()** `void EMF::EMRSAVEDC::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.39.3.2 serialize()** `bool EMF::EMRSAVEDC::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.39.3.3 size()** `int EMF::EMRSAVEDC::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

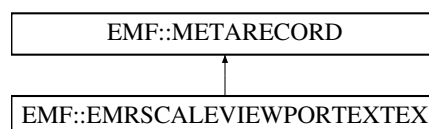
- libemf.h

## 3.40 EMF::EMRSCALEVIEWPORTEXTEX Class Reference

EMF Scale Viewport Extents (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSCALEVIEWPORTEXTEX:



## Public Member Functions

- [EMRSCALEVIEWPORTEXTEx](#) (LONG x\_num, LONG x\_den, LONG y\_num, LONG y\_den)
- [EMRSCALEVIEWPORTEXTEx](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.40.1 Detailed Description

EMF Scale Viewport Extents (ex)

The viewport extent is the device coordinate (i.e. pixels) size of the viewport. Scale the viewport extents by the ratios of the given values. (OpenOffice accepts this, but not SETVIEWPORTEXT(?))

### 3.40.2 Constructor & Destructor Documentation

**3.40.2.1 EMRSCALEVIEWPORTEXTEx() [1/2]** `EMF::EMRSCALEVIEWPORTEXTEx::EMRSCALEVIEWPORTEXTEx (`  
LONG x\_num,  
LONG x\_den,  
LONG y\_num,  
LONG y\_den ) [inline]

#### Parameters

<i>x_num</i>	numerator of x scale
<i>x_den</i>	denominator of x scale
<i>y_num</i>	numerator of y scale
<i>y_den</i>	denominator of y scale

**3.40.2.2 EMRSCALEVIEWPORTEXTEx() [2/2]** `EMF::EMRSCALEVIEWPORTEXTEx::EMRSCALEVIEWPORTEXTEx (`  
[DATASTREAM](#) & ds ) [inline]

Construct a ScaleViewportExtEx record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.40.3 Member Function Documentation



**3.40.3.1 execute()** `void EMF::EMRSCALEVIEWPORTEXTEX::execute (`  
    [METAFILEDEVICECONTEXT](#) \* *source*,  
    HDC *dc* ) const [inline], [virtual]

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.40.3.2 serialize()** `bool EMF::EMRSCALEVIEWPORTEXTEX::serialize (`  
    [DATASTREAM](#) *ds* ) [inline], [virtual]

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.40.3.3 size()** `int EMF::EMRSCALEVIEWPORTEXTEX::size (`  
    void ) const [inline], [virtual]

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

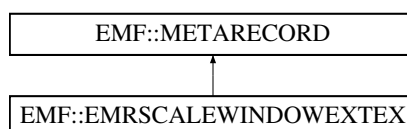
- `libemf.h`

## 3.41 EMF::EMRSCALEWINDOWEXTEx Class Reference

EMF Scale Window Extents (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSCALEWINDOWEXTEx:



## Public Member Functions

- [EMRSCALEWINDOWEXTEx](#) (LONG x\_num, LONG x\_den, LONG y\_num, LONG y\_den)
- [EMRSCALEWINDOWEXTEx](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.41.1 Detailed Description

EMF Scale Window Extents (ex)

The window extent is the logical coordinate size of the window. Scale the window extents by the ratios of the given values.

### 3.41.2 Constructor & Destructor Documentation

**3.41.2.1 EMRSCALEWINDOWEXTEx() [1/2]** `EMF::EMRSCALEWINDOWEXTEx::EMRSCALEWINDOWEXTEx (`  
`LONG x_num,`  
`LONG x_den,`  
`LONG y_num,`  
`LONG y_den ) [inline]`

#### Parameters

<i>x_num</i>	numerator of x scale
<i>x_den</i>	denominator of x scale
<i>y_num</i>	numerator of y scale
<i>y_den</i>	denominator of y scale

**3.41.2.2 EMRSCALEWINDOWEXTEx() [2/2]** `EMF::EMRSCALEWINDOWEXTEx::EMRSCALEWINDOWEXTEx (`  
`DATASTREAM & ds ) [inline]`

Construct a ScaleWindowExtEx record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.41.3 Member Function Documentation

**3.41.3.1 execute()** `void EMF::EMRSCALEWINDOWEXTEx::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.41.3.2 serialize()** `bool EMF::EMRSCALEWINDOWEXTEx::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.41.3.3 size()** `int EMF::EMRSCALEWINDOWEXTEx::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

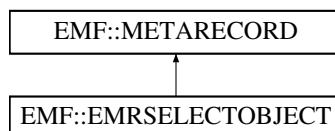
- libemf.h

## 3.42 EMF::EMRSELECTOBJECT Class Reference

EMF Select Object.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSELECTOBJECT:



## Public Member Functions

- [EMRSELECTOBJECT](#) (HGDIOBJ object)
- [EMRSELECTOBJECT](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.42.1 Detailed Description

EMF Select Object.

Activate (make current) the given object, such as a pen, brush or font.

### 3.42.2 Constructor & Destructor Documentation

**3.42.2.1 EMRSELECTOBJECT() [1/2]** `EMF::EMRSELECTOBJECT::EMRSELECTOBJECT ( HGDIOBJ object ) [inline]`

#### Parameters

<i>object</i>	the object to make active.
---------------	----------------------------

**3.42.2.2 EMRSELECTOBJECT() [2/2]** `EMF::EMRSELECTOBJECT::EMRSELECTOBJECT ( DATASTREAM & ds ) [inline]`

Construct a SelectObject record from the input stream.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.42.3 Member Function Documentation

**3.42.3.1 execute()** `void EMF::EMRSELECTOBJECT::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

References [EMF::METAFILEDEVICECONTEXT::emf\\_handles](#).

**3.42.3.2 serialize()** `bool EMF::EMRSELECTOBJECT::serialize (   
     DATASTREAM ds )   [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.42.3.3 size()** `int EMF::EMRSELECTOBJECT::size (   
     void ) const   [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following files:

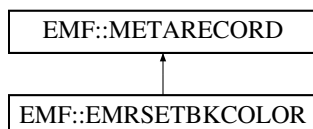
- libemf.h
- libemf.cpp

## 3.43 EMF::EMRSETBKCOLOR Class Reference

EMF Set Background Color.

```
#include <libemf.h>
```

Inheritance diagram for [EMF::EMRSETBKCOLOR](#):



## Public Member Functions

- [EMRSETBKCOLOR](#) (COLORREF color)
- [EMRSETBKCOLOR](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.43.1 Detailed Description

EMF Set Background Color.

Sets the background color.

### 3.43.2 Constructor & Destructor Documentation

**3.43.2.1 EMRSETBKCOLOR()** [1/2] `EMF::EMRSETBKCOLOR::EMRSETBKCOLOR ( COLORREF color ) [inline]`

Parameters

<i>color</i>	background color
--------------	------------------

**3.43.2.2 EMRSETBKCOLOR()** [2/2] `EMF::EMRSETBKCOLOR::EMRSETBKCOLOR ( DATASTREAM & ds ) [inline]`

Construct a SetBkColor record from the input datastream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.43.3 Member Function Documentation

**3.43.3.1 execute()** `void EMF::EMRSETBKCOLOR::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.43.3.2 serialize()** `bool EMF::EMRSETBKCOLOR::serialize (
 DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.43.3.3 size()** `int EMF::EMRSETBKCOLOR::size (
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

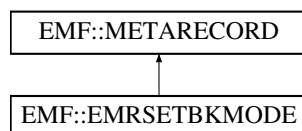
- libemf.h

**3.44 EMF::EMRSETBKMODE Class Reference**

EMF Set Background Mode.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETBKMODE:

**Public Member Functions**

- [EMRSETBKMODE](#) (DWORD mode)
- [EMRSETBKMODE](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.44.1 Detailed Description

EMF Set Background Mode.

Set the background mode: transparent or opaque. Seems to be ignored by StarOffice. (Appears to work for text, though.)

### 3.44.2 Constructor & Destructor Documentation

**3.44.2.1 EMRSETBKMODE()** [1/2] `EMF::EMRSETBKMODE::EMRSETBKMODE (   
 DWORD mode ) [inline]`

Parameters

<i>mode</i>	background mode.
-------------	------------------

**3.44.2.2 EMRSETBKMODE()** [2/2] `EMF::EMRSETBKMODE::EMRSETBKMODE (   
 DATASTREAM & ds ) [inline]`

Construct a SetBkMode record from the input datastream.

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.44.3 Member Function Documentation

**3.44.3.1 execute()** `void EMF::EMRSETBKMODE::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).



**3.44.3.2 serialize()** `bool EMF::EMRSETBKMODE::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.44.3.3 size()** `int EMF::EMRSETBKMODE::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

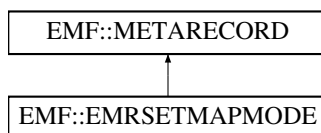
- libemf.h

## 3.45 EMF::EMRSETMAPMODE Class Reference

EMF Set Mapping Mode.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETMAPMODE:



### Public Member Functions

- [EMRSETMAPMODE](#) (DWORD mode)
- [EMRSETMAPMODE](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

### 3.45.1 Detailed Description

EMF Set Mapping Mode.

Set the mapping mode: HI (X style), LO (OpenGL style). Totally ignored by StarOffice as near as I can tell.

### 3.45.2 Constructor & Destructor Documentation

**3.45.2.1 EMRSETMAPMODE()** [1/2] `EMF::EMRSETMAPMODE::EMRSETMAPMODE (   
 DWORD mode ) [inline]`

## Parameters

<i>mode</i>	window mapping mode
-------------	---------------------

### 3.45.2.2 EMRSETMAPMODE() [2/2]

```
EMF::EMRSETMAPMODE::EMRSETMAPMODE (
    DATASTREAM & ds ) [inline]
```

Construct a SetMapMode record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.45.3 Member Function Documentation

### 3.45.3.1 execute()

```
void EMF::EMRSETMAPMODE::execute (
    METAFILEDEVICECONTEXT * source,
    HDC dc ) const [inline], [virtual]
```

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

### 3.45.3.2 serialize()

```
bool EMF::EMRSETMAPMODE::serialize (
    DATASTREAM ds ) [inline], [virtual]
```

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

### 3.45.3.3 size()

```
int EMF::EMRSETMAPMODE::size (
    void ) const [inline], [virtual]
```

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

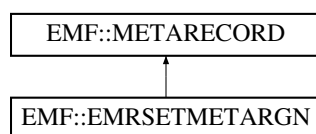
- libemf.h

### 3.46 EMF::EMRSETMETARGN Class Reference

EMF Set Meta Region.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETMETARGN:



#### Public Member Functions

- [EMRSETMETARGN](#) (void)
- [EMRSETMETARGN](#) ([DATASTREAM](#) &ds)
- bool [serialize](#) ([DATASTREAM](#) ds)
- int [size](#) (void) const
- void [execute](#) ([METAFILEDEVICECONTEXT](#) \*source, HDC dc) const

#### 3.46.1 Detailed Description

EMF Set Meta Region.

I really have no idea.

#### 3.46.2 Constructor & Destructor Documentation

**3.46.2.1 EMRSETMETARGN()** [1/2] `EMF::EMRSETMETARGN::EMRSETMETARGN (void) [inline]`

Create a Set Meta Rgn record.

**3.46.2.2 EMRSETMETARGN()** [2/2] `EMF::EMRSETMETARGN::EMRSETMETARGN (DATASTREAM & ds) [inline]`

Construct an Setmetargn record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.46.3 Member Function Documentation

**3.46.3.1 execute()** `void EMF::EMRSETMETARGN::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.46.3.2 serialize()** `bool EMF::EMRSETMETARGN::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.46.3.3 size()** `int EMF::EMRSETMETARGN::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

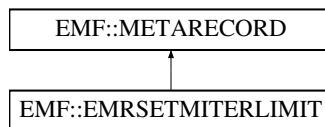
- `libemf.h`

### 3.47 EMF::EMRSETMITERLIMIT Class Reference

EMF SetMiterLimit.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETMITERLIMIT:



#### Public Member Functions

- [EMRSETMITERLIMIT](#) (FLOAT limit)
- [EMRSETMITERLIMIT](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.47.1 Detailed Description

EMF SetMiterLimit.

Sets the length limit for miter joins.

#### 3.47.2 Constructor & Destructor Documentation

**3.47.2.1 EMRSETMITERLIMIT()** [1/2] `EMF::EMRSETMITERLIMIT::EMRSETMITERLIMIT ( FLOAT limit ) [inline]`

Parameters

<i>limit</i>	miter length limit.
--------------	---------------------

**3.47.2.2 EMRSETMITERLIMIT()** [2/2] `EMF::EMRSETMITERLIMIT::EMRSETMITERLIMIT ( DATASTREAM & ds ) [inline]`

Construct a SetMiterLimit record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.47.3 Member Function Documentation

**3.47.3.1 execute()** `void EMF::EMRSETMITERLIMIT::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.47.3.2 serialize()** `bool EMF::EMRSETMITERLIMIT::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.47.3.3 size()** `int EMF::EMRSETMITERLIMIT::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

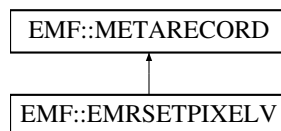
- [libemf.h](#)

### 3.48 EMF::EMRSETPIXELV Class Reference

EMF Set Pixel.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETPIXELV:



#### Public Member Functions

- [EMRSETPIXELV](#) (INT x, INT y, COLORREF color)
- [EMRSETPIXELV](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.48.1 Detailed Description

EMF Set Pixel.

Set the given pixel to the given color.

#### 3.48.2 Constructor & Destructor Documentation

**3.48.2.1 EMRSETPIXELV()** [1/2] `EMF::EMRSETPIXELV::EMRSETPIXELV (`  
     INT x,  
     INT y,  
     COLORREF color ) [inline]

##### Parameters

<i>x</i>	x position at which to draw pixel.
<i>y</i>	y position at which to draw pixel.
<i>color</i>	color of pixel.

**3.48.2.2 EMRSETPIXELV()** [2/2] `EMF::EMRSETPIXELV::EMRSETPIXELV (`  
     DATASTREAM & ds ) [inline]

Construct a SetPixelV record from the input stream.



**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.48.3 Member Function Documentation**

**3.48.3.1 execute()** `void EMF::EMRSETPIXELV::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.48.3.2 serialize()** `bool EMF::EMRSETPIXELV::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.48.3.3 size()** `int EMF::EMRSETPIXELV::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

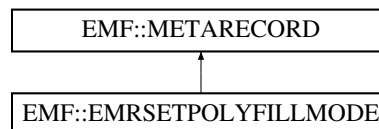
- [libemf.h](#)

## 3.49 EMF::EMRSETPOLYFILLMODE Class Reference

EMF Set the Polygon Fill Mode.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETPOLYFILLMODE:



### Public Member Functions

- [EMRSETPOLYFILLMODE](#) (DWORD mode)
- [EMRSETPOLYFILLMODE](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.49.1 Detailed Description

EMF Set the Polygon Fill Mode.

Set the polygon fill mode: ALTERNATE or WINDING

### 3.49.2 Constructor & Destructor Documentation

**3.49.2.1 EMRSETPOLYFILLMODE() [1/2]** `EMF::EMRSETPOLYFILLMODE::EMRSETPOLYFILLMODE (   
 DWORD mode ) [inline]`

Parameters

<i>mode</i>	background mode.
-------------	------------------

**3.49.2.2 EMRSETPOLYFILLMODE() [2/2]** `EMF::EMRSETPOLYFILLMODE::EMRSETPOLYFILLMODE (   
 DATASTREAM & ds ) [inline]`

Construct a SetPolyFillMode record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.49.3 Member Function Documentation**

**3.49.3.1 execute()** `void EMF::EMRSETPOLYFILLMODE::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.49.3.2 serialize()** `bool EMF::EMRSETPOLYFILLMODE::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.49.3.3 size()** `int EMF::EMRSETPOLYFILLMODE::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

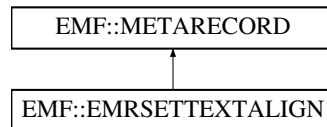
- libemf.h

### 3.50 EMF::EMRSETTEXTALIGN Class Reference

EMF Set Text Alignment.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETTEXTALIGN:



#### Public Member Functions

- [EMRSETTEXTALIGN](#) (UINT mode)
- [EMRSETTEXTALIGN](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.50.1 Detailed Description

EMF Set Text Alignment.

Determines the justification of the text with respect to its position.

#### 3.50.2 Constructor & Destructor Documentation

**3.50.2.1 EMRSETTEXTALIGN() [1/2]** `EMF::EMRSETTEXTALIGN::EMRSETTEXTALIGN (UINT mode) [inline]`

Parameters

<i>mode</i>	text alignment mode.
-------------	----------------------

**3.50.2.2 EMRSETTEXTALIGN() [2/2]** `EMF::EMRSETTEXTALIGN::EMRSETTEXTALIGN (DATASTREAM & ds) [inline]`

Construct a SetTextAlign record from the input datastream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.50.3 Member Function Documentation**

**3.50.3.1 execute()** `void EMF::EMRSETTEXTALIGN::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.50.3.2 serialize()** `bool EMF::EMRSETTEXTALIGN::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.50.3.3 size()** `int EMF::EMRSETTEXTALIGN::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

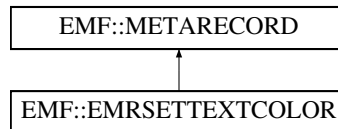
- [libemf.h](#)

## 3.51 EMF::EMRSETTEXTCOLOR Class Reference

EMF Set Text Color.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETTEXTCOLOR:



### Public Member Functions

- [EMRSETTEXTCOLOR](#) (COLORREF color)
- [EMRSETTEXTCOLOR](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.51.1 Detailed Description

EMF Set Text Color.

Sets the foreground color of text.

#### 3.51.2 Constructor & Destructor Documentation

**3.51.2.1 EMRSETTEXTCOLOR()** [1/2] `EMF::EMRSETTEXTCOLOR::EMRSETTEXTCOLOR ( COLORREF color ) [inline]`

Parameters

<i>color</i>	text foreground color
--------------	-----------------------

**3.51.2.2 EMRSETTEXTCOLOR()** [2/2] `EMF::EMRSETTEXTCOLOR::EMRSETTEXTCOLOR ( DATASTREAM & ds ) [inline]`

Construct a SetTextColor record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.51.3 Member Function Documentation**

**3.51.3.1 execute()** `void EMF::EMRSETTEXTCOLOR::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.51.3.2 serialize()** `bool EMF::EMRSETTEXTCOLOR::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.51.3.3 size()** `int EMF::EMRSETTEXTCOLOR::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

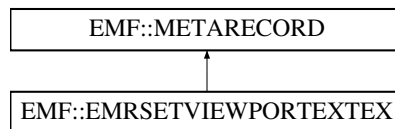
- [libemf.h](#)

## 3.52 EMF::EMRSETVIEWPORTETEX Class Reference

EMF Set Viewport Extents (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETVIEWPORTETEX:



### Public Member Functions

- [EMRSETVIEWPORTETEX](#) (INT cx, INT cy)
- [EMRSETVIEWPORTETEX](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

### 3.52.1 Detailed Description

EMF Set Viewport Extents (ex)

The viewport extent is the device coordinate (i.e. pixels) size of the viewport. Since W32 doesn't do any clipping, the purpose of this is not clear.

### 3.52.2 Constructor & Destructor Documentation

**3.52.2.1 EMRSETVIEWPORTETEX() [1/2]** `EMF::EMRSETVIEWPORTETEX::EMRSETVIEWPORTETEX ( INT cx, INT cy ) [inline]`

Parameters

<i>cx</i>	width of viewport in device coordinates
<i>cy</i>	height of viewport in device coordinates

**3.52.2.2 EMRSETVIEWPORTETEX() [2/2]** `EMF::EMRSETVIEWPORTETEX::EMRSETVIEWPORTETEX ( DATASTREAM & ds ) [inline]`

Construct a SetViewportExtEx record from the input stream.



**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.52.3 Member Function Documentation**

**3.52.3.1 execute()** `void EMF::EMRSETVIEWPORTEXTEX::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.52.3.2 serialize()** `bool EMF::EMRSETVIEWPORTEXTEX::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.52.3.3 size()** `int EMF::EMRSETVIEWPORTEXTEX::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

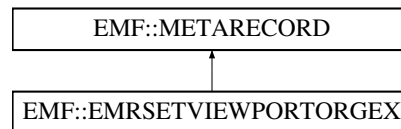
- libemf.h

### 3.53 EMF::EMRSETVIEWPORTORGEX Class Reference

EMF Set Viewport Origin (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETVIEWPORTORGEX:



#### Public Member Functions

- [EMRSETVIEWPORTORGEX](#) (INT x, INT y)
- [EMRSETVIEWPORTORGEX](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.53.1 Detailed Description

EMF Set Viewport Origin (ex)

The viewport origin is a point in device coordinates (i.e., pixels) where the viewport starts. (For example, if you want to put several different views on the same page, you might use different viewports.)

#### 3.53.2 Constructor & Destructor Documentation

**3.53.2.1 EMRSETVIEWPORTORGEX()** [1/2] `EMF::EMRSETVIEWPORTORGEX::EMRSETVIEWPORTORGEX (`  
     INT x,  
     INT y ) [inline]

##### Parameters

x	x position of the viewport in device coordinates
y	y position of the viewport in device coordinates

**3.53.2.2 EMRSETVIEWPORTORGEX()** [2/2] `EMF::EMRSETVIEWPORTORGEX::EMRSETVIEWPORTORGEX (`  
     DATASTREAM & ds ) [inline]

Construct a SetVieportOrgEx record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.53.3 Member Function Documentation**

**3.53.3.1 execute()** `void EMF::EMRSETVIEWPORTORGEX::execute (   
    METAFILEDEVICECONTEXT * source,   
    HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.53.3.2 serialize()** `bool EMF::EMRSETVIEWPORTORGEX::serialize (   
    DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.53.3.3 size()** `int EMF::EMRSETVIEWPORTORGEX::size (   
    void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

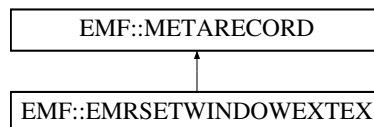
- libemf.h

### 3.54 EMF::EMRSETWINDOWEXTEx Class Reference

EMF Set Window Extent (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETWINDOWEXTEx:



#### Public Member Functions

- [EMRSETWINDOWEXTEx](#) (INT cx, INT cy)
- [EMRSETWINDOWEXTEx](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.54.1 Detailed Description

EMF Set Window Extent (ex)

The window extents define the scale of the logical coordinates. For example, if your XY plot is from [-10,-10] to [10,10], then the window extents are [20,20].

#### 3.54.2 Constructor & Destructor Documentation

**3.54.2.1 EMRSETWINDOWEXTEx() [1/2]** `EMF::EMRSETWINDOWEXTEx::EMRSETWINDOWEXTEx (`  
     INT cx,  
     INT cy ) `[inline]`

Parameters

cx	width of window in logical coordinates.
cy	height of window in logical coordinates.

**3.54.2.2 EMRSETWINDOWEXTEx() [2/2]** `EMF::EMRSETWINDOWEXTEx::EMRSETWINDOWEXTEx (`  
     DATASTREAM & ds ) `[inline]`

Construct a SetWindowExtEx record from the input stream.

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

**3.54.3 Member Function Documentation**

**3.54.3.1 execute()** `void EMF::EMRSETWINDOWEXTEx::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

**Parameters**

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.54.3.2 serialize()** `bool EMF::EMRSETWINDOWEXTEx::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

**Parameters**

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.54.3.3 size()** `int EMF::EMRSETWINDOWEXTEx::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

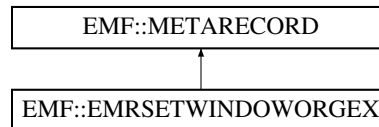
- libemf.h

### 3.55 EMF::EMRSETWINDOWORGEX Class Reference

EMF Set Window Origin (ex)

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETWINDOWORGEX:



#### Public Member Functions

- [EMRSETWINDOWORGEX](#) (INT x, INT y)
- [EMRSETWINDOWORGEX](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.55.1 Detailed Description

EMF Set Window Origin (ex)

The window origin specifies the logical (i.e., real) coordinates of the upper, left corner of the viewport. (For example, if you want your XY plot's axis origin to be in the middle of the viewport, you'd set the window origin to something like [-1,-1].)

#### 3.55.2 Constructor & Destructor Documentation

**3.55.2.1 EMRSETWINDOWORGEX()** [1/2] `EMF::EMRSETWINDOWORGEX::EMRSETWINDOWORGEX (`  
     INT x,  
     INT y ) [inline]

##### Parameters

x	x coordinate of window origin in logical coordinates
y	y coordinate of window origin in logical coordinates

**3.55.2.2 EMRSETWINDOWORGEX()** [2/2] `EMF::EMRSETWINDOWORGEX::EMRSETWINDOWORGEX (`  
     DATASTREAM & ds ) [inline]

Construct a SetWindowOrgEx record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.55.3 Member Function Documentation

**3.55.3.1 execute()** `void EMF::EMRSETWINDOWORGE::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.55.3.2 serialize()** `bool EMF::EMRSETWINDOWORGE::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.55.3.3 size()** `int EMF::EMRSETWINDOWORGE::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

- `libemf.h`

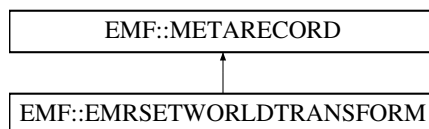


### 3.56 EMF::EMRSETWORLDTRANSFORM Class Reference

EMF Set World Transform.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSETWORLDTRANSFORM:



#### Public Member Functions

- [EMRSETWORLDTRANSFORM](#) (const XFORM \*transform)
- [EMRSETWORLDTRANSFORM](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.56.1 Detailed Description

EMF Set World Transform.

Enhanced metafiles have a Coordinate Transformation which allows the contents to be rotated and transformed. Does not appear to work properly in StarOffice (but it's also possible I don't understand how it's supposed to work either).

#### 3.56.2 Constructor & Destructor Documentation

**3.56.2.1 EMRSETWORLDTRANSFORM()** [1/2] EMF::EMRSETWORLDTRANSFORM::EMRSETWORLDTRANSFORM ( const XFORM \* *transform* ) [inline]

Parameters

<i>transform</i>	the new transformation
------------------	------------------------

**3.56.2.2 EMRSETWORLDTRANSFORM()** [2/2] EMF::EMRSETWORLDTRANSFORM::EMRSETWORLDTRANSFORM ( DATASTREAM & *ds* ) [inline]

Construct a SetWorldTransform record from the input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

### 3.56.3 Member Function Documentation

**3.56.3.1 execute()** `void EMF::EMRSETWORLDTRANSFORM::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.56.3.2 serialize()** `bool EMF::EMRSETWORLDTRANSFORM::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.56.3.3 size()** `int EMF::EMRSETWORLDTRANSFORM::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

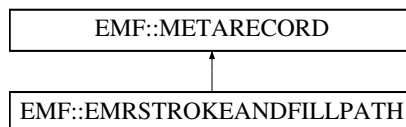
- libemf.h

### 3.57 EMF::EMRSTROKEANDFILLPATH Class Reference

EMF Stroke and Fill path.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSTROKEANDFILLPATH:



#### Public Member Functions

- [EMRSTROKEANDFILLPATH](#) (const RECTL \*bounds)
- [EMRSTROKEANDFILLPATH](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.57.1 Detailed Description

EMF Stroke and Fill path.

Stroke and Fill the path.

#### 3.57.2 Constructor & Destructor Documentation

**3.57.2.1 EMRSTROKEANDFILLPATH()** [1/2] `EMF::EMRSTROKEANDFILLPATH::EMRSTROKEANDFILLPATH (const RECTL * bounds) [inline]`

Parameters

<i>bounds</i>	overall bounding box of polygon.
---------------	----------------------------------

**3.57.2.2 EMRSTROKEANDFILLPATH()** [2/2] `EMF::EMRSTROKEANDFILLPATH::EMRSTROKEANDFILLPATH (DATASTREAM & ds) [inline]`

Create a StrokeandfillPath record from input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.57.3 Member Function Documentation

**3.57.3.1 execute()** `void EMF::EMRSTROKEANDFILLPATH::execute (   
METAFILEDEVICECONTEXT * source,   
HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.57.3.2 serialize()** `bool EMF::EMRSTROKEANDFILLPATH::serialize (   
DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.57.3.3 size()** `int EMF::EMRSTROKEANDFILLPATH::size (   
void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

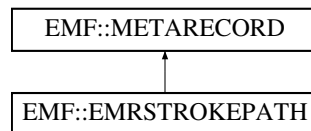
- libemf.h

### 3.58 EMF::EMRSTROKEPATH Class Reference

EMF Stroke path.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EMRSTROKEPATH:



#### Public Member Functions

- [EMRSTROKEPATH](#) (const RECTL \*bounds)
- [EMRSTROKEPATH](#) (DATASTREAM &ds)
- bool [serialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.58.1 Detailed Description

EMF Stroke path.

Stroke the path.

#### 3.58.2 Constructor & Destructor Documentation

**3.58.2.1 EMRSTROKEPATH()** [1/2] `EMF::EMRSTROKEPATH::EMRSTROKEPATH (const RECTL * bounds ) [inline]`

Parameters

<i>bounds</i>	overall bounding box of polygon.
---------------	----------------------------------

**3.58.2.2 EMRSTROKEPATH()** [2/2] `EMF::EMRSTROKEPATH::EMRSTROKEPATH (DATASTREAM & ds ) [inline]`

Create a StrokePath record from input stream.

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

## 3.58.3 Member Function Documentation

**3.58.3.1 execute()** `void EMF::EMRSTROKEPATH::execute (   
METAFILEDEVICECONTEXT * source,   
HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

## Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.58.3.2 serialize()** `bool EMF::EMRSTROKEPATH::serialize (   
DATASTREAM ds ) [inline], [virtual]`

## Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.58.3.3 size()** `int EMF::EMRSTROKEPATH::size (   
void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

The documentation for this class was generated from the following file:

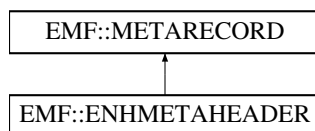
- `libemf.h`

### 3.59 EMF::ENHMETAHEADER Class Reference

Enhanced Metafile Header Record.

```
#include <libemf.h>
```

Inheritance diagram for EMF::ENHMETAHEADER:



#### Public Member Functions

- [ENHMETAHEADER](#) (LPCWSTR description=0)
- [~ENHMETAHEADER](#) ()
- bool [serialize](#) (DATASTREAM ds)
- bool [unserialize](#) (DATASTREAM ds)
- int [size](#) (void) const
- void [execute](#) (METAFILEDEVICECONTEXT \*source, HDC dc) const

#### 3.59.1 Detailed Description

Enhanced Metafile Header Record.

The [ENHMETAHEADER](#) serves two purposes in this library: it keeps track of the size of the metafile (in physical dimensions) and number of records and handles that are ultimately to be written to the disk file. It is also a real record that must be written out.

#### 3.59.2 Constructor & Destructor Documentation

**3.59.2.1 ENHMETAHEADER()** `EMF::ENHMETAHEADER::ENHMETAHEADER ( LPCWSTR description = 0 ) [inline]`

##### Parameters

<i>description</i>	an optional description argument is a UNICODE-like string with the following format: "some text\0some more text\0\0". The W32 interface defines UNICODE characters to be two-byte (unsigned short strings). The constructor makes a copy of the argument.
--------------------	---

**3.59.2.2 ~ENHMETAHEADER()** `EMF::ENHMETAHEADER::~~ENHMETAHEADER ( ) [inline]`

Destructor deletes memory allocated for description.

### 3.59.3 Member Function Documentation

**3.59.3.1 execute()** `void EMF::ENHMETAHEADER::execute (   
 METAFILEDEVICECONTEXT * source,   
 HDC dc ) const [inline], [virtual]`

Execute this record in the context of the given device context.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	device context for execute.

Implements [EMF::METARECORD](#).

**3.59.3.2 serialize()** `bool EMF::ENHMETAHEADER::serialize (   
 DATASTREAM ds ) [inline], [virtual]`

Serializing the header is an example of an extended record.

#### Parameters

<i>ds</i>	Metafile datastream.
-----------	----------------------

Implements [EMF::METARECORD](#).

**3.59.3.3 size()** `int EMF::ENHMETAHEADER::size (   
 void ) const [inline], [virtual]`

Internally computed size of this record.

Implements [EMF::METARECORD](#).

**3.59.3.4 unserialize()** `bool EMF::ENHMETAHEADER::unserialize (   
 DATASTREAM ds ) [inline]`

Read a header record from the datastream.

The documentation for this class was generated from the following file:

- libemf.h

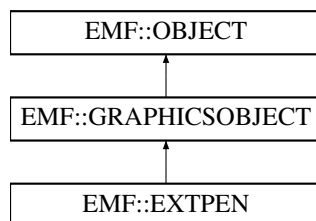


### 3.60 EMF::EXTPEN Class Reference

Extended Graphics Pen.

```
#include <libemf.h>
```

Inheritance diagram for EMF::EXTPEN:



#### Public Member Functions

- [EXTPEN](#) (const EXTLOGPEN \**lpen*)
- OBJECTTYPE [getType](#) (void) const
- METARECORD \* [newEMR](#) (HDC dc, HGDIOBJ emf\_handle)

#### Additional Inherited Members

##### 3.60.1 Detailed Description

Extended Graphics Pen.

Pens are used for drawing lines, arc, rectangles, etc.

##### 3.60.2 Constructor & Destructor Documentation

**3.60.2.1** **EXTPEN()** `EMF::EXTPEN::EXTPEN (const EXTLOGPEN * lpen) [inline]`

#### Parameters

<i>lpen</i>	the (logical?) pen definition.
-------------	--------------------------------

##### 3.60.3 Member Function Documentation

**3.60.3.1** `getType()` `OBJECTTYPE EMF::EXTPEN::getType (`  
`void ) const [inline], [virtual]`

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.60.3.2** `newEMR()` `METARECORD* EMF::EXTPEN::newEMR (`  
`HDC dc,`  
`HGDIOBJ emf_handle ) [inline], [virtual]`

Return a new metarecord for this object. And record its selection into the given device context.

#### Parameters

<i>dc</i>	handle of device context into which this object is being selected.
<i>emf_handle</i>	the EMF handle associated with the <a href="#">PEN</a> .

Implements [EMF::GRAPHICSOBJECT](#).

References [EMF::GRAPHICSOBJECT::contexts](#).

The documentation for this class was generated from the following file:

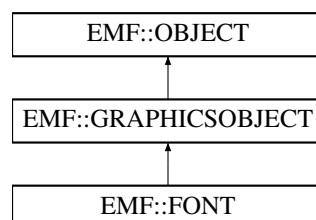
- [libemf.h](#)

## 3.61 EMF::FONT Class Reference

Graphics Font.

```
#include <libemf.h>
```

Inheritance diagram for [EMF::FONT](#):



#### Public Member Functions

- [FONT](#) (const LOGFONTW \*lfont)
- `OBJECTTYPE` [getType](#) (void) const
- `METARECORD *` [newEMR](#) (HDC dc, HGDIOBJ emf\_handle)

## Additional Inherited Members

### 3.61.1 Detailed Description

Graphics Font.

Fonts are used for drawing text (obviously).

### 3.61.2 Constructor & Destructor Documentation

**3.61.2.1 FONT()** `EMF::FONT::FONT (const LOGFONTW * lfont ) [inline]`

#### Parameters

<i>lfont</i>	the (logical?) font definition.
--------------	---------------------------------

### 3.61.3 Member Function Documentation

**3.61.3.1 getType()** `OBJECTTYPE EMF::FONT::getType (void ) const [inline], [virtual]`

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.61.3.2 newEMR()** `METARECORD* EMF::FONT::newEMR (HDC dc, HGDIOBJ emf_handle ) [inline], [virtual]`

Return a new metarecord for this object. And record its selection into the given device context.

#### Parameters

<i>dc</i>	handle of device context into which this object is being selected.
<i>emf_handle</i>	the EMF handle associated with the <a href="#">FONT</a> .

Implements [EMF::GRAPHICSOBJECT](#).

References [EMF::GRAPHICSOBJECT::contexts](#).

The documentation for this class was generated from the following file:

- libemf.h

## 3.62 EMF::GLOBALOBJECTS Class Reference

```
#include <libemf.h>
```

### Public Member Functions

- HGDIOBJ [add](#) (OBJECT \*object)
- OBJECT \* [find](#) (const HGDIOBJ handle)
- void [remove](#) (const OBJECT \*object)
- auto [begin](#) (void) const
- auto [end](#) (void) const
- METARECORDCTOR [newRecord](#) (DWORD iType) const

### Static Public Member Functions

- static EMF::METARECORD \* [new\\_eof](#) (DATASTREAM &ds)  
*Create a new EMREOF record.*
- static EMF::METARECORD \* [new\\_setviewportorgex](#) (DATASTREAM &ds)  
*Create a new EMRSETVIEWPORTORGEX record.*
- static EMF::METARECORD \* [new\\_setwindoworgex](#) (DATASTREAM &ds)  
*Create a new EMRSETWINDOWORGEX record.*
- static EMF::METARECORD \* [new\\_setviewporttex](#) (DATASTREAM &ds)  
*Create a new EMRSETVIEWPORTEX record.*
- static EMF::METARECORD \* [new\\_setwindowextex](#) (DATASTREAM &ds)  
*Create a new EMRSETWINDOWEXTEX record.*
- static EMF::METARECORD \* [new\\_scaleviewportextex](#) (DATASTREAM &ds)  
*Create a new SCALEVIEWPORTEX record.*
- static EMF::METARECORD \* [new\\_scalewindowextex](#) (DATASTREAM &ds)  
*Create a new SCALEWINDOWEXTEX record.*
- static EMF::METARECORD \* [new\\_modifyworldtransform](#) (DATASTREAM &ds)  
*Create a new MODIFYWORLDTRANSFORM record.*
- static EMF::METARECORD \* [new\\_setworldtransform](#) (DATASTREAM &ds)  
*Create a new SETWORLDTRANSFORM record.*
- static EMF::METARECORD \* [new\\_settextalign](#) (DATASTREAM &ds)  
*Create a new SETTEXTALIGN record.*
- static EMF::METARECORD \* [new\\_settextcolor](#) (DATASTREAM &ds)  
*Create a new SETTEXTCOLOR record.*
- static EMF::METARECORD \* [new\\_setbkcolor](#) (DATASTREAM &ds)  
*Create a new SETBKCOLOR record.*
- static EMF::METARECORD \* [new\\_setbkmode](#) (DATASTREAM &ds)  
*Create a new SETBKMODE record.*
- static EMF::METARECORD \* [new\\_setpolyfillmode](#) (DATASTREAM &ds)  
*Create a new SETPOLYFILLMODE record.*
- static EMF::METARECORD \* [new\\_setmapmode](#) (DATASTREAM &ds)  
*Create a new SETMAPMODE record.*

- static `EMF::METARECORD * new_selectobject (DATASTREAM &ds)`  
*Create a new SELECTOBJECT record.*
- static `EMF::METARECORD * new_deleteobject (DATASTREAM &ds)`  
*Create a new DELETEOBJECT record.*
- static `EMF::METARECORD * new_movetoex (DATASTREAM &ds)`  
*Create a new MOVETOEX record.*
- static `EMF::METARECORD * new_lineto (DATASTREAM &ds)`  
*Create a new LINETO record.*
- static `EMF::METARECORD * new_arc (DATASTREAM &ds)`  
*Create a new ARC record.*
- static `EMF::METARECORD * new_arcto (DATASTREAM &ds)`  
*Create a new ARCTO record.*
- static `EMF::METARECORD * new_rectangle (DATASTREAM &ds)`  
*Create a new RECTANGLE record.*
- static `EMF::METARECORD * new_ellipse (DATASTREAM &ds)`  
*Create a new ELLIPSE record.*
- static `EMF::METARECORD * new_polyline (DATASTREAM &ds)`  
*Create a new POLYLINE record.*
- static `EMF::METARECORD * new_polyline16 (DATASTREAM &ds)`  
*Create a new POLYLINE16 record.*
- static `EMF::METARECORD * new_polygon (DATASTREAM &ds)`  
*Create a new POLYGON record.*
- static `EMF::METARECORD * new_polygon16 (DATASTREAM &ds)`  
*Create a new POLYGON16 record.*
- static `EMF::METARECORD * new_polypolygon (DATASTREAM &ds)`  
*Create a new POLYPOLYGON record.*
- static `EMF::METARECORD * new_polypolygon16 (DATASTREAM &ds)`  
*Create a new POLYPOLYGON16 record.*
- static `EMF::METARECORD * new_polybezier (DATASTREAM &ds)`  
*Create a new POLYBEZIER record.*
- static `EMF::METARECORD * new_polybezier16 (DATASTREAM &ds)`  
*Create a new POLYBEZIER16 record.*
- static `EMF::METARECORD * new_polybezierto (DATASTREAM &ds)`  
*Create a new POLYBEZIERTO record.*
- static `EMF::METARECORD * new_polybezierto16 (DATASTREAM &ds)`  
*Create a new POLYBEZIERTO16 record.*
- static `EMF::METARECORD * new_polylineto (DATASTREAM &ds)`  
*Create a new POLYLINETO record.*
- static `EMF::METARECORD * new_polylineto16 (DATASTREAM &ds)`  
*Create a new POLYLINETO16 record.*
- static `EMF::METARECORD * new_exttextouta (DATASTREAM &ds)`  
*Create a new EXTTEXTOUTA record.*
- static `EMF::METARECORD * new_exttextoutw (DATASTREAM &ds)`  
*Create a new EXTTEXTOUTW record.*
- static `EMF::METARECORD * new_setpixelv (DATASTREAM &ds)`  
*Create a new SETPIXELV record.*
- static `EMF::METARECORD * new_createpen (DATASTREAM &ds)`  
*Create a new CREATEPEN record.*
- static `EMF::METARECORD * new_extcreatepen (DATASTREAM &ds)`  
*Create a new EXTCREATEPEN record.*
- static `EMF::METARECORD * new_createbrushindirect (DATASTREAM &ds)`

- Create a new CREATEBRUSHINDIRECT record.*

  - static `EMF::METARECORD * new_extcreatefontindirectw (DATASTREAM &ds)`

*Create a new EXTCREATEFONTINDIRECTW record.*
- static `EMF::METARECORD * new_fillpath (DATASTREAM &ds)`

*Create a new FILLPATH record.*
- static `EMF::METARECORD * new_strokepath (DATASTREAM &ds)`

*Create a new STROKEPATH record.*
- static `EMF::METARECORD * new_strokeandfillpath (DATASTREAM &ds)`

*Create a new STROKEANDFILLPATH record.*
- static `EMF::METARECORD * new_beginpath (DATASTREAM &ds)`

*Create a new BEGINPATH record.*
- static `EMF::METARECORD * new_endpath (DATASTREAM &ds)`

*Create a new ENDPATH record.*
- static `EMF::METARECORD * new_closefigure (DATASTREAM &ds)`

*Create a new CLOSEFIGURE record.*
- static `EMF::METARECORD * new_savedc (DATASTREAM &ds)`

*Create a new SAVEDC record.*
- static `EMF::METARECORD * new_restoredc (DATASTREAM &ds)`

*Create a new RESTOREDC record.*
- static `EMF::METARECORD * new_setmetargn (DATASTREAM &ds)`

*Create a new SETMETARGN record.*
- static `EMF::METARECORD * new_setmiterlimit (DATASTREAM &ds)`

*Create a new SETMITERLIMIT record.*

### 3.62.1 Detailed Description

Stores all the objects in a single database within a process.

### 3.62.2 Member Function Documentation

**3.62.2.1 add()** `HGDIOBJ EMF::GLOBALOBJECTS::add ( OBJECT * object )`

Add an object to the global vector. The object's handle is simply its index in the global object vector, which is computed by the very interesting "difference between two iterators" method.

#### Parameters

<code>object</code>	pointer to a real instance of an object, not its handle.
---------------------	--

**3.62.2.2 begin()** `auto EMF::GLOBALOBJECTS::begin ( void ) const [inline]`

**Returns**

an iterator pointing to the first global object.

**3.62.2.3 end()** `auto EMF::GLOBALOBJECTS::end (`  
`void ) const [inline]`

**Returns**

an iterator pointing to (one past) the final global object.

**3.62.2.4 find()** `OBJECT * EMF::GLOBALOBJECTS::find (`  
`const HGDIOBJ handle )`

Look up a object by handle in the global object vector. Note: Stock objects (like a gray brush or the black pen) have their high order bit set, so this has to be masked out when using their handles.

**Parameters**

<i>handle</i>	the object's handle.
---------------	----------------------

**Returns**

pointer to object.

**3.62.2.5 newRecord()** `METARECORDCTOR EMF::GLOBALOBJECTS::newRecord (`  
`DWORD iType ) const`

See if we have a constructor for a record of the given type.

**Parameters**

<i>iType</i>	metarecord type.
--------------	------------------

**Returns**

pointer to "virtual" constructor.

**3.62.2.6 remove()** `void EMF::GLOBALOBJECTS::remove (`  
`const OBJECT * object )`

A call to the metafile function `DeleteObject()` allows a particular object's handle to be reused, so some care has to be taken to erase it.



## Parameters

<i>object</i>	pointer to object to delete.
---------------	------------------------------

The documentation for this class was generated from the following files:

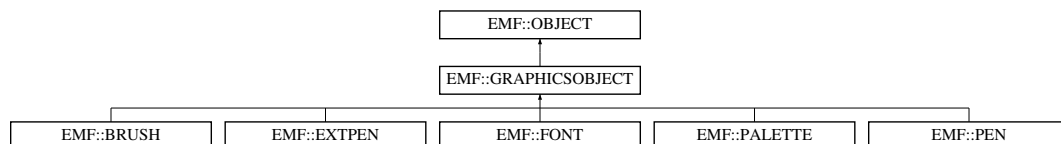
- libemf.h
- libemf.cpp

### 3.63 EMF::GRAPHICSOBJECT Class Reference

A global graphics object.

```
#include <libemf.h>
```

Inheritance diagram for EMF::GRAPHICSOBJECT:



#### Public Member Functions

- virtual [~GRAPHICSOBJECT](#) ()  
*GRAPHICSOBJECTs has a virtual destructor.*
- virtual [METARECORD](#) \* [newEMR](#) (HDC dc, HGDIOBJ [handle](#))=0

#### Data Fields

- `std::map< HDC, HGDIOBJ >` [contexts](#)

#### 3.63.1 Detailed Description

A global graphics object.

Graphics objects have some additional properties: When an object is Select'ed into a device context, the handle for that context is added to the list of context's in which this object is used.

#### 3.63.2 Member Function Documentation

**3.63.2.1 newEMR()** `virtual METARECORD* EMF::GRAPHICSOBJECT::newEMR (`  
     HDC *dc*,  
     HGDIOBJ *handle* ) `[pure virtual]`

Create a new metarecord which describes this object.

## Parameters

<i>dc</i>	the handle to the device context.
<i>handle</i>	(appears not to used. Note the handle is really assigned at serialization time.)

Implemented in [EMF::PALETTE](#), [EMF::FONT](#), [EMF::BRUSH](#), [EMF::EXTPEN](#), and [EMF::PEN](#).

## 3.63.3 Field Documentation

3.63.3.1 contexts `std::map< HDC, HGDIOBJ > EMF::GRAPHICSOBJECT::contexts`

A set of all the contexts into which this object has been selected and the associated metafile handle for the object.

Referenced by [EMF::PEN::newEMR\(\)](#), [EMF::EXTPEN::newEMR\(\)](#), [EMF::BRUSH::newEMR\(\)](#), [EMF::FONT::newEMR\(\)](#), and [EMF::PALETTE::newEMR\(\)](#).

The documentation for this class was generated from the following file:

- `libemf.h`

## 3.64 EMF::INTARRAY Struct Reference

Represent an array of integers in a simple way.

```
#include <libemf.h>
```

## Public Member Functions

- [INTARRAY](#) ([INT](#) \*const *ints*, const [DWORD](#) *n*)

## Data Fields

- [INT](#) \*const [ints\\_](#)  
*Array of ints.*
- const [DWORD](#) [n\\_](#)  
*Number of ints in array.*

## 3.64.1 Detailed Description

Represent an array of integers in a simple way.

Allow an array of [INT](#)'s to be written out at once.

## 3.64.2 Constructor &amp; Destructor Documentation

3.64.2.1 [INTARRAY\(\)](#) `EMF::INTARRAY::INTARRAY (`  

```

    INT *const ints,
    const DWORD n ) [inline]
```

simple constructor.

## Parameters

<i>ints</i>	pointer to ints.
<i>n</i>	number ints in array.

The documentation for this struct was generated from the following file:

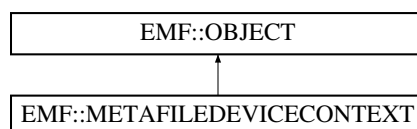
- libemf.h

### 3.65 EMF::METAFILEDEVICECONTEXT Class Reference

Graphics Device Context.

```
#include <libemf.h>
```

Inheritance diagram for EMF::METAFILEDEVICECONTEXT:



#### Public Member Functions

- [METAFILEDEVICECONTEXT](#) (FILE \*fp\_, const RECT \*size, LPCWSTR description\_w)
- virtual [~METAFILEDEVICECONTEXT](#) ()
- OBJECTTYPE [getType](#) (void) const
- DWORD [nextHandle](#) (void)
- void [clearHandle](#) (DWORD handle)
- void [appendRecord](#) (METARECORD \*record)
- void [appendHandle](#) (METARECORD \*record)
- void [deleteMetafile](#) (void)
- void [mergePoint](#) (const LONG &x, const LONG &y)
- void [mergePoint](#) (const POINT &p)

#### Data Fields

- ::FILE \* [fp](#)
- DATASTREAM [ds](#)
- ENHMETAHEADER \* [header](#)
- std::vector< [EMF::METARECORD](#) \* > [records](#)
- SIZEL [resolution](#)  
*The resolution in DPI of the reference DC.*
- SIZEL [viewport\\_ext](#)  
*The extent of the viewport.*
- POINT [viewport\\_org](#)  
*The origin of the viewport.*
- SIZEL [window\\_ext](#)

- The extent of the window.*
  - POINT [window\\_org](#)
- The origin of the window.*
  - bool [update\\_frame](#)
- Update the frame automatically?*
  - POINT [min\\_device\\_point](#)
- The lft/top-most painted point in device units.*
  - POINT [max\\_device\\_point](#)
- The rgt/btm-most painted point in device units.*
  - POINT [point](#)
- The current point.*
  - PEN \* [pen](#)
- The current pen.*
  - BRUSH \* [brush](#)
- The current brush.*
  - FONT \* [font](#)
- The current font.*
  - PALETTE \* [palette](#)
- The current palette.*
  - UINT [text\\_alignment](#)
- The current text alignment.*
  - COLORREF [text\\_color](#)
- The current text foreground color.*
  - COLORREF [bk\\_color](#)
- The current background color.*
  - INT [bk\\_mode](#)
- The current background mode.*
  - INT [polyfill\\_mode](#)
- The current polygon fill mode.*
  - INT [map\\_mode](#)
- The current mapping mode.*
  - FLOAT [miter\\_limit](#)
- The current miter length limit.*
  - std::vector< bool > [handles](#)
  - std::map< HGDIOBJ, HGDIOBJ > [emf\\_handles](#)

### 3.65.1 Detailed Description

Graphics Device Context.

Almost all GDI graphics calls require a device context (except those which create graphics objects such as pens and fonts). This is a specific context which renders to a metafile. There is a one-to-one correspondence between the device context and the metafile.

### 3.65.2 Constructor & Destructor Documentation

**3.65.2.1 METAFILEDEVICECONTEXT()** `EMF::METAFILEDEVICECONTEXT::METAFILEDEVICECONTEXT ( FILE * fp_, const RECT * size, LPCWSTR description_w ) [inline]`

Most graphics programs seem to want to handle the opening and closing of files themselves, so this is an extension to the w32 interface.

## Parameters

<i>fp_</i>	stdio pointer to an open file. May be null.
<i>size</i>	the rectangle describing the position and size of the metafile on the "page". May be null.
<i>description_↵ _w</i>	a UNICODE string describing the metafile. The format must be "some text\0some more text\0\0". May be null.

**3.65.2.2 ~METAFILEDEVICECONTEXT()** `virtual EMF::METAFILEDEVICECONTEXT::~~METAFILEDEVICECONTEXT ( ) [inline], [virtual]`

Destructor frees all the graphics objects which may have been allocated. Now, it also frees any metarecords which it might hold, too.

References deleteMetafile(), and records.

### 3.65.3 Member Function Documentation

**3.65.3.1 appendHandle()** `void EMF::METAFILEDEVICECONTEXT::appendHandle ( METARECORD * record ) [inline]`

Add this record to the metafile.

## Parameters

<i>record</i>	this record is an object so it increments the handle count as well.
---------------	---

References header, records, and EMF::METARECORD::size().

**3.65.3.2 appendRecord()** `void EMF::METAFILEDEVICECONTEXT::appendRecord ( METARECORD * record ) [inline]`

Add this record to the metafile.

## Parameters

<i>record</i>	standard graphics record
---------------	--------------------------

References header, records, and EMF::METARECORD::size().

**3.65.3.3 clearHandle()** `void EMF::METAFILEDEVICECONTEXT::clearHandle (   
 DWORD handle ) [inline]`

Clear the usage of this handle

References EMF::OBJECT::handle, and handles.

**3.65.3.4 deleteMetafile()** `void EMF::METAFILEDEVICECONTEXT::deleteMetafile (   
 void ) [inline]`

Delete all the records from the metafile. This would seem to include deleting the header record as well.

References records.

Referenced by ~METAFILEDEVICECONTEXT().

**3.65.3.5 getType()** `OBJECTTYPE EMF::METAFILEDEVICECONTEXT::getType (   
 void ) const [inline], [virtual]`

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.65.3.6 mergePoint() [1/2]** `void EMF::METAFILEDEVICECONTEXT::mergePoint (   
 const LONG & x,   
 const LONG & y ) [inline]`

Somewhat superfluous, except checker doesn't understand the initialization of automatic structures in the declaration.

**3.65.3.7 mergePoint() [2/2]** `void EMF::METAFILEDEVICECONTEXT::mergePoint (   
 const POINT & p ) [inline]`

Take the given point and determine if it enlarges the "painted" area of the device.

References header, max\_device\_point, min\_device\_point, update\_frame, viewport\_ext, viewport\_org, window\_ext, and window\_org.

**3.65.3.8 nextHandle()** `DWORD EMF::METAFILEDEVICECONTEXT::nextHandle (   
 void ) [inline]`

Scan the bit vector of used handles and return the index of the first free bit as this objects metafile handle.

References handles, and header.

### 3.65.4 Field Documentation

#### 3.65.4.1 **ds** `DATASTREAM` `EMF::METAFILEDEVICECONTEXT::ds`

All i/o to the metafile is wrapped by this class so that byte swapping on big-endian machines is transparent.

#### 3.65.4.2 **emf\_handles** `std::map< HGDIOBJ, HGDIOBJ > EMF::METAFILEDEVICECONTEXT::emf_handles`

This map holds the *current* mapping between EMF handles and global object handles as a metafile is played back (with `PlayEnhMetaFile`).

Referenced by `EMF::EMRSELECTOBJECT::execute()`, `EMF::EMRDELETEOBJECT::execute()`, `EMF::EMRCREATEPEN::execute()`, `EMF::EMREXTCREATEPEN::execute()`, `EMF::EMRCREATEBRUSHINDIRECT::execute()`, and `EMF::EMREXTCREATEFONTINDIRECTW::execute()`.

#### 3.65.4.3 **fp** `::FILE*` `EMF::METAFILEDEVICECONTEXT::fp`

If it is a file-based metafile, then this pointer is not null.

#### 3.65.4.4 **handles** `std::vector< bool > EMF::METAFILEDEVICECONTEXT::handles`

For compatibility, it appears that metafile handles are reused as objects are deleted. Attempt to emulate that behavior with a bit vector of used metafile handles.

Referenced by `clearHandle()`, and `nextHandle()`.

#### 3.65.4.5 **header** `ENHMETAHEADER*` `EMF::METAFILEDEVICECONTEXT::header`

Serves double duty as the physical device description.

Referenced by `appendHandle()`, `appendRecord()`, `mergePoint()`, and `nextHandle()`.

#### 3.65.4.6 **records** `std::vector< EMF::METARECORD* > EMF::METAFILEDEVICECONTEXT::records`

All of the metafile records are stored in memory.

Referenced by `appendHandle()`, `appendRecord()`, `deleteMetafile()`, and `~METAFILEDEVICECONTEXT()`.

The documentation for this class was generated from the following file:

- `libemf.h`





## 3.66.2 Constructor & Destructor Documentation

### 3.66.2.1 `~METARECORD()` `virtual EMF::METARECORD::~METARECORD ( ) [inline], [virtual]`

The virtual destructor allows records which allocated additional memory to release it when they are deleted. Simple records just use the default destructor defined here.

## 3.66.3 Member Function Documentation

### 3.66.3.1 `execute()` `virtual void EMF::METARECORD::execute ( METAFILEDEVICECONTEXT * source, HDC dc ) const [pure virtual]`

Execute the graphics command in the given context. Used by PlayEnhMetaFile to "copy" one metafile into another.

#### Parameters

<i>source</i>	the device context from which this record is taken.
<i>dc</i>	the destination context.

Implemented in [EMF::EMRSETMITERLIMIT](#), [EMF::EMRSETMETARGN](#), [EMF::EMRRESTORED](#), [EMF::EMRSAVE](#), [EMF::EMRCLOSEFIGURE](#), [EMF::EMRENDPATH](#), [EMF::EMRBEGINPATH](#), [EMF::EMRSTROKEANDFILLPATH](#), [EMF::EMRSTROKEPATH](#), [EMF::EMRFILLPATH](#), [EMF::EMRCREATEPALETTE](#), [EMF::EMREXTCREATEFONTINDIRECTW](#), [EMF::EMRCREATEBRUSHINDIRECT](#), [EMF::EMREXTCREATEPEN](#), [EMF::EMRCREATEPEN](#), [EMF::EMRSETPIXELV](#), [EMF::EMREXTTEXTOUTW](#), [EMF::EMREXTTEXTOUTA](#), [EMF::EMRPOLYLINETO16](#), [EMF::EMRPOLYLINETO](#), [EMF::EMRPOLYBEZIERTO16](#), [EMF::EMRPOLYBEZIERTO](#), [EMF::EMRPOLYBEZIER16](#), [EMF::EMRPOLYBEZIER](#), [EMF::EMRPOLYPOLYGON16](#), [EMF::EMRPOLYPOLYGON](#), [EMF::EMRPOLYGON16](#), [EMF::EMRPOLYGON](#), [EMF::EMRPOLYLINE16](#), [EMF::EMRPOLYLINE](#), [EMF::EMRELLIPSE](#), [EMF::EMRRECTANGLE](#), [EMF::EMRARCTO](#), [EMF::EMRARC](#), [EMF::EMRLINETO](#), [EMF::EMRMOVETOEX](#), [EMF::EMRDELETEOBJECT](#), [EMF::EMRSELECTOBJECT](#), [EMF::EMRSETMAPMODE](#), [EMF::EMRSETPOLYFILLMODE](#), [EMF::EMRSETBKMODE](#), [EMF::EMRSETBKCOLOR](#), [EMF::EMRSETTEXTCOLOR](#), [EMF::EMRSETTEXTALIGN](#), [EMF::EMRSETWORLDTRANSFORM](#), [EMF::EMRMODIFYWORLDTRANSFORM](#), [EMF::EMRSCALEWINDOWEXTEXT](#), [EMF::EMRSETWINDOWEXTEXT](#), [EMF::EMRSCALEVIEWPORTEXTEXT](#), [EMF::EMRSETVIEWPORTEXTEXT](#), [EMF::EMRSETWINDOWORGEX](#), [EMF::EMRSETVIEWPORTORGEX](#), [EMF::EMREOF](#), and [EMF::ENHMETAHEADER](#).

### 3.66.3.2 `serialize()` `virtual bool EMF::METARECORD::serialize ( DATASTREAM ds ) [pure virtual]`

Write yourself to the given file. This is virtual since some records are of arbitrary length and need to write additional information after their EMR structure.

#### Parameters

<i>ds</i>	the datastream to write oneself to.
-----------	-------------------------------------

Implemented in [EMF::EMRSETMITERLIMIT](#), [EMF::EMRSETMETARGN](#), [EMF::EMRRESTOREDC](#), [EMF::EMRSAVEDC](#), [EMF::EMRCLOSEFIGURE](#), [EMF::EMRENDPATH](#), [EMF::EMRBEGINPATH](#), [EMF::EMRSTROKEANDFILLPATH](#), [EMF::EMRSTROKEPATH](#), [EMF::EMRFILLPATH](#), [EMF::EMRCREATEPALETTE](#), [EMF::EMREXTCREATEFONTINDIRECTW](#), [EMF::EMRCREATEBRUSHINDIRECT](#), [EMF::EMREXTCREATEPEN](#), [EMF::EMRCREATEPEN](#), [EMF::EMRSETPIXELV](#), [EMF::EMREXTTEXTOUTW](#), [EMF::EMREXTTEXTOUTA](#), [EMF::EMRPOLYLINETO16](#), [EMF::EMRPOLYLINETO](#), [EMF::EMRPOLYBEZIERTO16](#), [EMF::EMRPOLYBEZIERTO](#), [EMF::EMRPOLYBEZIER16](#), [EMF::EMRPOLYBEZIER](#), [EMF::EMRPOLYPOLYGON16](#), [EMF::EMRPOLYPOLYGON](#), [EMF::EMRPOLYGON16](#), [EMF::EMRPOLYGON](#), [EMF::EMRPOLYLINE16](#), [EMF::EMRPOLYLINE](#), [EMF::EMRELLIPSE](#), [EMF::EMRRECTANGLE](#), [EMF::EMRARCTO](#), [EMF::EMRARC](#), [EMF::EMRLINETO](#), [EMF::EMRMOVETOEX](#), [EMF::EMRDELETEOBJECT](#), [EMF::EMRSELECTOBJECT](#), [EMF::EMRSETMAPMODE](#), [EMF::EMRSETPOLYFILLMODE](#), [EMF::EMRSETBKMODE](#), [EMF::EMRSETBKCOLOR](#), [EMF::EMRSETTEXTCOLOR](#), [EMF::EMRSETTEXTALIGN](#), [EMF::EMRSETWORLDTRANSFORM](#), [EMF::EMRMODIFYWORLDTRANSFORM](#), [EMF::EMRSCALEWINDOWEXTEX](#), [EMF::EMRSETWINDOWEXTEX](#), [EMF::EMRSCALEVIEWPORTEXTEX](#), [EMF::EMRSETVIEWPORTEXTEX](#), [EMF::EMRSETWINDOWORGEX](#), [EMF::EMRSETVIEWPORTORGEX](#), [EMF::EMREOF](#), and [EMF::ENHMETAHEADER](#).

**3.66.3.3 size()** `virtual int EMF::METARECORD::size (void) const [pure virtual]`

The header record of a metafile records the total size of the metafile in bytes, so as each record is added to the list, it updates the total size.

Implemented in [EMF::EMRSETMITERLIMIT](#), [EMF::EMRSETMETARGN](#), [EMF::EMRRESTOREDC](#), [EMF::EMRSAVEDC](#), [EMF::EMRCLOSEFIGURE](#), [EMF::EMRENDPATH](#), [EMF::EMRBEGINPATH](#), [EMF::EMRSTROKEANDFILLPATH](#), [EMF::EMRSTROKEPATH](#), [EMF::EMRFILLPATH](#), [EMF::EMRCREATEPALETTE](#), [EMF::EMREXTCREATEFONTINDIRECTW](#), [EMF::EMRCREATEBRUSHINDIRECT](#), [EMF::EMREXTCREATEPEN](#), [EMF::EMRCREATEPEN](#), [EMF::EMRSETPIXELV](#), [EMF::EMREXTTEXTOUTW](#), [EMF::EMREXTTEXTOUTA](#), [EMF::EMRPOLYLINETO16](#), [EMF::EMRPOLYLINETO](#), [EMF::EMRPOLYBEZIERTO16](#), [EMF::EMRPOLYBEZIERTO](#), [EMF::EMRPOLYBEZIER16](#), [EMF::EMRPOLYBEZIER](#), [EMF::EMRPOLYPOLYGON16](#), [EMF::EMRPOLYPOLYGON](#), [EMF::EMRPOLYGON16](#), [EMF::EMRPOLYGON](#), [EMF::EMRPOLYLINE16](#), [EMF::EMRPOLYLINE](#), [EMF::EMRELLIPSE](#), [EMF::EMRRECTANGLE](#), [EMF::EMRARCTO](#), [EMF::EMRARC](#), [EMF::EMRLINETO](#), [EMF::EMRMOVETOEX](#), [EMF::EMRDELETEOBJECT](#), [EMF::EMRSELECTOBJECT](#), [EMF::EMRSETMAPMODE](#), [EMF::EMRSETPOLYFILLMODE](#), [EMF::EMRSETBKMODE](#), [EMF::EMRSETBKCOLOR](#), [EMF::EMRSETTEXTCOLOR](#), [EMF::EMRSETTEXTALIGN](#), [EMF::EMRSETWORLDTRANSFORM](#), [EMF::EMRMODIFYWORLDTRANSFORM](#), [EMF::EMRSCALEWINDOWEXTEX](#), [EMF::EMRSETWINDOWEXTEX](#), [EMF::EMRSCALEVIEWPORTEXTEX](#), [EMF::EMRSETVIEWPORTEXTEX](#), [EMF::EMRSETWINDOWORGEX](#), [EMF::EMRSETVIEWPORTORGEX](#), [EMF::EMREOF](#), and [EMF::ENHMETAHEADER](#).

Referenced by [EMF::METAFILEDEVICECONTEXT::appendHandle\(\)](#), and [EMF::METAFILEDEVICECONTEXT::appendRecord\(\)](#).

The documentation for this class was generated from the following file:

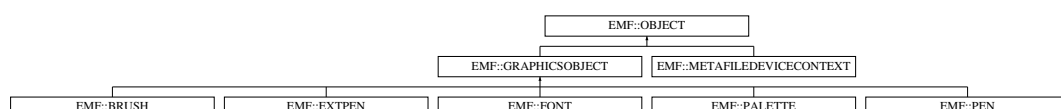
- `libemf.h`

## 3.67 EMF::OBJECT Class Reference

Global GDI object.

```
#include <libemf.h>
```

Inheritance diagram for EMF::OBJECT:



## Public Member Functions

- virtual [~OBJECT](#) ()  
*OBJECTs have a virtual destructor.*
- [OBJECT](#) (void)
- virtual OBJECTTYPE [getType](#) (void) const =0

## Data Fields

- HGDIOBJ [handle](#)

### 3.67.1 Detailed Description

Global GDI object.

The GDI interface defines objects in terms of handles (rather than pointers). In order to emulate this, each object is placed into a global list and the index in that list becomes their handle.

### 3.67.2 Constructor & Destructor Documentation

**3.67.2.1 [OBJECT\(\)](#)** `EMF::OBJECT::OBJECT (`  
`void ) [inline]`

Create a new object. It's up to a subclass to create a real handle for this object.

### 3.67.3 Member Function Documentation

**3.67.3.1 [getType\(\)](#)** `virtual OBJECTTYPE EMF::OBJECT::getType (`  
`void ) const [pure virtual]`

Return the type of the object.

Implemented in [EMF::METAFILEDEVICECONTEXT](#), [EMF::PALETTE](#), [EMF::FONT](#), [EMF::BRUSH](#), [EMF::EXTPEN](#), and [EMF::PEN](#).

### 3.67.4 Field Documentation

#### 3.67.4.1 handle `HGDIOBJ EMF::OBJECT::handle`

The handle of a GDI object.

Referenced by `EMF::METAFILEDEVICECONTEXT::clearHandle()`.

The documentation for this class was generated from the following file:

- `libemf.h`

## 3.68 EMF::PADDING Struct Reference

All metafile records must be padded out to a multiple of 4 bytes.

```
#include <libemf.h>
```

### Public Member Functions

- [PADDING](#) (const int size)

### Data Fields

- const int [size\\_](#)  
*Number of bytes of padding.*

### Static Public Attributes

- static const char [padding\\_](#) [4] = { 0, 0, 0, 0 }  
*Pad with '0's.*

#### 3.68.1 Detailed Description

All metafile records must be padded out to a multiple of 4 bytes.

Write out a few bytes of padding if necessary.

#### 3.68.2 Constructor & Destructor Documentation

##### 3.68.2.1 PADDING() `EMF::PADDING::PADDING (const int size) [inline]`

simple constructor.

## Parameters

<i>size</i>	number of bytes of padding to use.
-------------	------------------------------------

The documentation for this struct was generated from the following files:

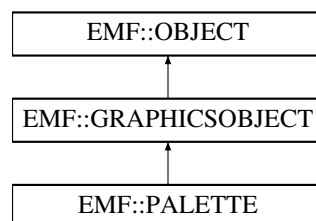
- libemf.h
- libemf.cpp

### 3.69 EMF::PALETTE Class Reference

Graphics Palette.

```
#include <libemf.h>
```

Inheritance diagram for EMF::PALETTE:



#### Public Member Functions

- [PALETTE](#) (const LOGPALETTE \*lpalette)
- OBJECTTYPE [getType](#) (void) const
- [METARECORD](#) \* [newEMR](#) (HDC dc, HGDIOBJ emf\_handle)

#### Additional Inherited Members

##### 3.69.1 Detailed Description

Graphics Palette.

Not entirely sure how palettes are used in general.

##### 3.69.2 Constructor & Destructor Documentation

**3.69.2.1 PALETTE()** `EMF::PALETTE::PALETTE (const LOGPALETTE * lpalette) [inline]`

## Parameters

<i>palette</i>	the (logical?) palette definition.
----------------	------------------------------------

## 3.69.3 Member Function Documentation

**3.69.3.1 `getType()`** OBJECTTYPE EMF::PALETTE::getType ( void ) const [inline], [virtual]

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.69.3.2 `newEMR()`** METARECORD\* EMF::PALETTE::newEMR ( HDC *dc*, HGDIOBJ *emf\_handle* ) [inline], [virtual]

Return a new metarecord for this object. And record its selection into the given device context.

## Parameters

<i>dc</i>	handle of device context into which this object is being selected.
<i>emf_handle</i>	the EMF handle associated with the <a href="#">FONT</a> .

Implements [EMF::GRAPHICSOBJECT](#).

References EMF::GRAPHICSOBJECT::contexts.

The documentation for this class was generated from the following file:

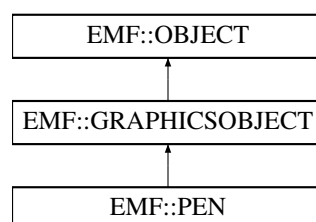
- libemf.h

## 3.70 EMF::PEN Class Reference

Graphics Pen.

```
#include <libemf.h>
```

Inheritance diagram for EMF::PEN:



## Public Member Functions

- [PEN](#) (const LOGPEN \*lpen)
- OBJECTTYPE [getType](#) (void) const
- [METARECORD](#) \* [newEMR](#) (HDC dc, HGDIOBJ emf\_handle)

## Additional Inherited Members

### 3.70.1 Detailed Description

Graphics Pen.

Pens are used for drawing lines, arc, rectangles, etc.

### 3.70.2 Constructor & Destructor Documentation

**3.70.2.1 PEN()** `EMF::PEN::PEN (const LOGPEN * lpen) [inline]`

#### Parameters

<i>lpen</i>	the (logical?) pen definition.
-------------	--------------------------------

### 3.70.3 Member Function Documentation

**3.70.3.1 getType()** `OBJECTTYPE EMF::PEN::getType (void) const [inline], [virtual]`

Return the type of this object (could probably do better with RTTI()).

Implements [EMF::OBJECT](#).

**3.70.3.2 newEMR()** `METARECORD* EMF::PEN::newEMR (HDC dc, HGDIOBJ emf_handle) [inline], [virtual]`

Return a new metarecord for this object. And record its selection into the given device context.

#### Parameters

<i>dc</i>	handle of device context into which this object is being selected.
<i>emf_handle</i>	the EMF handle associated with the <a href="#">PEN</a> .

Implements [EMF::GRAPHICSOBJECT](#).

References [EMF::GRAPHICSOBJECT::contexts](#).

The documentation for this class was generated from the following file:

- [libemf.h](#)

## 3.71 EMF::POINT16ARRAY Struct Reference

Represent an array of 16-bit point in a simple way.

```
#include <libemf.h>
```

### Public Member Functions

- [POINT16ARRAY](#) ([POINT16](#) \*const *points*, const [DWORD](#) *n*)

### Data Fields

- [POINT16](#) \*const [points\\_](#)  
*Array of POINT16s.*
- const [DWORD](#) [n\\_](#)  
*Number of POINT16s in array.*

### 3.71.1 Detailed Description

Represent an array of 16-bit point in a simple way.

Allow an array of POINT16's to be written out at once.

### 3.71.2 Constructor & Destructor Documentation

**3.71.2.1 POINT16ARRAY()** [EMF::POINT16ARRAY::POINT16ARRAY](#) (  
    [POINT16](#) \*const *points*,  
    const [DWORD](#) *n* ) [inline]

Simple constructor.

#### Parameters

<i>points</i>	pointer to array of POINT16s.
<i>n</i>	number POINT16s in array.



The documentation for this struct was generated from the following file:

- `libemf.h`

## 3.72 EMF::POINTLARRAY Struct Reference

Represent an array of points in a simple way.

```
#include <libemf.h>
```

### Public Member Functions

- [POINTLARRAY](#) ([POINTL](#) \*const *points*, const `DWORD` *n*)

### Data Fields

- `POINTL` \*const [points\\_](#)  
*Array of POINTLs.*
- const `DWORD` [n\\_](#)  
*Number of POINTLs in array.*

### 3.72.1 Detailed Description

Represent an array of points in a simple way.

Allow an array of `POINTL`'s to be written out at once.

### 3.72.2 Constructor & Destructor Documentation

**3.72.2.1 POINTLARRAY()** `EMF::POINTLARRAY::POINTLARRAY (`  
    `POINTL *const points,`  
    `const DWORD n ) [inline]`

Simple constructor.

#### Parameters

<i>points</i>	pointer to array of <code>POINTL</code> s.
<i>n</i>	number <code>POINTL</code> s in array.

The documentation for this struct was generated from the following file:

- `libemf.h`

## 3.73 EMF::WCHARSTR Struct Reference

Represent a wide (UNICODE) character string in a simple way.

```
#include <libemf.h>
```

### Public Member Functions

- [WCHARSTR](#) (WCHAR \*const string, const int length)

### Data Fields

- WCHAR \*const [string\\_](#)  
*String of WCHARs.*
- const int [length\\_](#)  
*Number of WCHARs in string.*

### 3.73.1 Detailed Description

Represent a wide (UNICODE) character string in a simple way.

Even (wchar) strings have to be byte swapped. This structure allows us to provide a uniform stream-like interface for writing out all the components of metafiles.

### 3.73.2 Constructor & Destructor Documentation

**3.73.2.1 WCHARSTR()** `EMF::WCHARSTR::WCHARSTR (`  
    WCHAR \*const *string*,  
    const int *length* ) `[inline]`

Simple constructor.

#### Parameters

<i>string</i>	pointer to string of WCHARs.
<i>length</i>	number of WCHARs in string.

The documentation for this struct was generated from the following file:

- libemf.h



## Index

- ~EMREXTTEXTOUTA
  - EMF::EMREXTTEXTOUTA, [57](#)
- ~EMREXTTEXTOUTW
  - EMF::EMREXTTEXTOUTW, [59](#)
- ~EMRPOLYBEZIER
  - EMF::EMRPOLYBEZIER, [69](#)
- ~EMRPOLYBEZIER16
  - EMF::EMRPOLYBEZIER16, [71](#)
- ~EMRPOLYBEZIERTO
  - EMF::EMRPOLYBEZIERTO, [74](#)
- ~EMRPOLYBEZIERTO16
  - EMF::EMRPOLYBEZIERTO16, [77](#)
- ~EMRPOLYGON
  - EMF::EMRPOLYGON, [79](#)
- ~EMRPOLYGON16
  - EMF::EMRPOLYGON16, [81](#)
- ~EMRPOLYLINE
  - EMF::EMRPOLYLINE, [84](#)
- ~EMRPOLYLINE16
  - EMF::EMRPOLYLINE16, [87](#)
- ~EMRPOLYLINETO
  - EMF::EMRPOLYLINETO, [89](#)
- ~EMRPOLYLINETO16
  - EMF::EMRPOLYLINETO16, [91](#)
- ~EMRPOLYPOLYGON
  - EMF::EMRPOLYPOLYGON, [94](#)
- ~EMRPOLYPOLYGON16
  - EMF::EMRPOLYPOLYGON16, [97](#)
- ~ENHMETAHEADER
  - EMF::ENHMETAHEADER, [143](#)
- ~METAFILEDEVICECONTEXT
  - EMF::METAFILEDEVICECONTEXT, [157](#)
- ~METARECORD
  - EMF::METARECORD, [161](#)
- add
  - EMF::GLOBALOBJECTS, [150](#)
- appendHandle
  - EMF::METAFILEDEVICECONTEXT, [157](#)
- appendRecord
  - EMF::METAFILEDEVICECONTEXT, [157](#)
- begin
  - EMF::GLOBALOBJECTS, [150](#)
- BRUSH
  - EMF::BRUSH, [7](#)
- BYTEARRAY
  - EMF::BYTEARRAY, [9](#)
- CHARSTR
  - EMF::CHARSTR, [10](#)
- clearHandle
  - EMF::METAFILEDEVICECONTEXT, [157](#)
- contexts
  - EMF::GRAPHICSOBJECT, [154](#)
- DATASTREAM
  - EMF::DATASTREAM, [12](#)
- deleteMetafile
  - EMF::METAFILEDEVICECONTEXT, [158](#)
- ds
  - EMF::METAFILEDEVICECONTEXT, [159](#)
- DWORDARRAY
  - EMF::DWORDARRAY, [29](#)
- EMF::BRUSH, [7](#)
  - BRUSH, [7](#)
  - getType, [8](#)
  - newEMR, [8](#)
- EMF::BYTEARRAY, [8](#)
  - BYTEARRAY, [9](#)
- EMF::CHARSTR, [9](#)
  - CHARSTR, [10](#)
- EMF::DATASTREAM, [10](#)
  - DATASTREAM, [12](#)
  - operator<<, [12–17](#), [19](#), [20](#)
  - operator>>, [20–27](#)
  - setStream, [28](#)
- EMF::DWORDARRAY, [28](#)
  - DWORDARRAY, [29](#)
- EMF::EMRARC, [29](#)
  - EMRARC, [30](#)
  - execute, [30](#)
  - serialize, [31](#)
  - size, [31](#)
- EMF::EMRARCTO, [31](#)
  - EMRARCTO, [32](#)
  - execute, [32](#)
  - serialize, [33](#)
  - size, [33](#)
- EMF::EMRBEGINPATH, [33](#)
  - EMRBEGINPATH, [34](#)
  - execute, [34](#)
  - serialize, [35](#)
  - size, [35](#)
- EMF::EMRCLOSEFIGURE, [35](#)
  - EMRCLOSEFIGURE, [36](#)
  - execute, [36](#)
  - serialize, [36](#)
  - size, [37](#)
- EMF::EMRCREATEBRUSHINDIRECT, [37](#)
  - EMRCREATEBRUSHINDIRECT, [37](#), [38](#)
  - execute, [38](#)
  - serialize, [38](#)
  - size, [38](#)
- EMF::EMRCREATEPALETTE, [39](#)
  - EMRCREATEPALETTE, [39](#), [40](#)
  - execute, [40](#)
  - serialize, [40](#)
  - size, [40](#)
- EMF::EMRCREATEPEN, [41](#)

- EMRCREATEPEN, 41, 42
  - execute, 42
  - serialize, 42
  - size, 42
- EMF::EMRDELETEOBJECT, 43
  - EMRDELETEOBJECT, 43, 44
  - execute, 44
  - serialize, 44
  - size, 44
- EMF::EMRELLIPSE, 45
  - EMRELLIPSE, 45, 46
  - execute, 46
  - serialize, 46
  - size, 46
- EMF::EMRENDPATH, 47
  - EMRENDPATH, 47
  - execute, 48
  - serialize, 48
  - size, 48
- EMF::EMREOF, 49
  - EMREOF, 49
  - execute, 50
  - serialize, 50
  - size, 50
- EMF::EMREXTCREATEFONTINDIRECTW, 50
  - EMREXTCREATEFONTINDIRECTW, 51
  - execute, 51
  - serialize, 52
  - size, 52
- EMF::EMREXTCREATEPEN, 52
  - EMREXTCREATEPEN, 53
  - execute, 53
  - serialize, 55
  - size, 55
- EMF::EMREXTTEXTOUTA, 55
  - ~EMREXTTEXTOUTA, 57
  - EMREXTTEXTOUTA, 56
  - execute, 57
  - serialize, 57
  - size, 57
- EMF::EMREXTTEXTOUTW, 58
  - ~EMREXTTEXTOUTW, 59
  - EMREXTTEXTOUTW, 58, 59
  - execute, 59
  - serialize, 59
  - size, 59
- EMF::EMRFILLPATH, 60
  - EMRFILLPATH, 60, 61
  - execute, 61
  - serialize, 61
  - size, 61
- EMF::EMRLINETO, 62
  - EMRLINETO, 62, 63
  - execute, 63
  - serialize, 63
  - size, 63
- EMF::EMRMODIFYWORLDTRANSFORM, 64
  - EMRMODIFYWORLDTRANSFORM, 64
  - execute, 65
  - serialize, 65
  - size, 65
- EMF::EMRMOVETOEX, 66
  - EMRMOVETOEX, 66
  - execute, 67
  - serialize, 67
  - size, 67
- EMF::EMRPOLYBEZIER, 68
  - ~EMRPOLYBEZIER, 69
  - EMRPOLYBEZIER, 68
  - execute, 69
  - serialize, 69
  - size, 69
- EMF::EMRPOLYBEZIER16, 70
  - ~EMRPOLYBEZIER16, 71
  - EMRPOLYBEZIER16, 70, 71
  - execute, 71
  - serialize, 73
  - size, 73
- EMF::EMRPOLYBEZIERTO, 73
  - ~EMRPOLYBEZIERTO, 74
  - EMRPOLYBEZIERTO, 74
  - execute, 75
  - serialize, 75
  - size, 75
- EMF::EMRPOLYBEZIERTO16, 75
  - ~EMRPOLYBEZIERTO16, 77
  - EMRPOLYBEZIERTO16, 76, 77
  - execute, 77
  - serialize, 77
  - size, 77
- EMF::EMRPOLYGON, 78
  - ~EMRPOLYGON, 79
  - EMRPOLYGON, 78, 79
  - execute, 79
  - serialize, 79
  - size, 80
- EMF::EMRPOLYGON16, 80
  - ~EMRPOLYGON16, 81
  - EMRPOLYGON16, 80, 81
  - execute, 81
  - serialize, 83
  - size, 83
- EMF::EMRPOLYLINE, 83
  - ~EMRPOLYLINE, 84
  - EMRPOLYLINE, 84
  - execute, 85
  - serialize, 85
  - size, 85
- EMF::EMRPOLYLINE16, 85
  - ~EMRPOLYLINE16, 87
  - EMRPOLYLINE16, 86, 87
  - execute, 87
  - serialize, 87
  - size, 87
- EMF::EMRPOLYLINETO, 88
  - ~EMRPOLYLINETO, 89

- EMRPOLYLINETO, [88](#), [89](#)
  - execute, [89](#)
  - serialize, [89](#)
  - size, [90](#)
- EMF::EMRPOLYLINETO16, [90](#)
  - ~EMRPOLYLINETO16, [91](#)
  - EMRPOLYLINETO16, [90](#), [91](#)
  - execute, [91](#)
  - serialize, [93](#)
  - size, [93](#)
- EMF::EMRPOLYPOLYGON, [93](#)
  - ~EMRPOLYPOLYGON, [94](#)
  - EMRPOLYPOLYGON, [94](#)
  - execute, [95](#)
  - serialize, [95](#)
  - size, [95](#)
- EMF::EMRPOLYPOLYGON16, [95](#)
  - ~EMRPOLYPOLYGON16, [97](#)
  - EMRPOLYPOLYGON16, [96](#), [97](#)
  - execute, [97](#)
  - serialize, [97](#)
  - size, [97](#)
- EMF::EMRRECTANGLE, [98](#)
  - EMRRECTANGLE, [98](#), [99](#)
  - execute, [99](#)
  - serialize, [99](#)
  - size, [99](#)
- EMF::EMRRESTOREDC, [100](#)
  - EMRRESTOREDC, [100](#)
  - execute, [101](#)
  - serialize, [101](#)
  - size, [101](#)
- EMF::EMRSAVEDC, [102](#)
  - EMRSAVEDC, [102](#)
  - execute, [103](#)
  - serialize, [103](#)
  - size, [103](#)
- EMF::EMRSCALEVIEWPORTEXT, [103](#)
  - EMRSCALEVIEWPORTEXT, [104](#)
  - execute, [104](#)
  - serialize, [105](#)
  - size, [105](#)
- EMF::EMRSCALEWINDOWEXT, [105](#)
  - EMRSCALEWINDOWEXT, [106](#)
  - execute, [106](#)
  - serialize, [107](#)
  - size, [107](#)
- EMF::EMRSELECTOBJECT, [107](#)
  - EMRSELECTOBJECT, [108](#)
  - execute, [108](#)
  - serialize, [109](#)
  - size, [109](#)
- EMF::EMRSETBKCOLOR, [109](#)
  - EMRSETBKCOLOR, [110](#)
  - execute, [110](#)
  - serialize, [111](#)
  - size, [111](#)
- EMF::EMRSETBKMODE, [111](#)
  - EMRSETBKMODE, [112](#)
  - execute, [112](#)
  - serialize, [112](#)
  - size, [113](#)
- EMF::EMRSETMAPMODE, [113](#)
  - EMRSETMAPMODE, [113](#), [114](#)
  - execute, [114](#)
  - serialize, [114](#)
  - size, [114](#)
- EMF::EMRSETMETARGN, [115](#)
  - EMRSETMETARGN, [115](#)
  - execute, [116](#)
  - serialize, [116](#)
  - size, [116](#)
- EMF::EMRSETMITERLIMIT, [117](#)
  - EMRSETMITERLIMIT, [117](#)
  - execute, [118](#)
  - serialize, [118](#)
  - size, [118](#)
- EMF::EMRSETPIXELV, [119](#)
  - EMRSETPIXELV, [119](#)
  - execute, [121](#)
  - serialize, [121](#)
  - size, [121](#)
- EMF::EMRSETPOLYFILLMODE, [122](#)
  - EMRSETPOLYFILLMODE, [122](#)
  - execute, [123](#)
  - serialize, [123](#)
  - size, [123](#)
- EMF::EMRSETTEXTALIGN, [124](#)
  - EMRSETTEXTALIGN, [124](#)
  - execute, [125](#)
  - serialize, [125](#)
  - size, [125](#)
- EMF::EMRSETTEXTCOLOR, [126](#)
  - EMRSETTEXTCOLOR, [126](#)
  - execute, [127](#)
  - serialize, [127](#)
  - size, [127](#)
- EMF::EMRSETVIEWPORTEXT, [128](#)
  - EMRSETVIEWPORTEXT, [128](#)
  - execute, [129](#)
  - serialize, [129](#)
  - size, [129](#)
- EMF::EMRSETVIEWPORTORGEX, [130](#)
  - EMRSETVIEWPORTORGEX, [130](#)
  - execute, [131](#)
  - serialize, [131](#)
  - size, [131](#)
- EMF::EMRSETWINDOWEXT, [132](#)
  - EMRSETWINDOWEXT, [132](#)
  - execute, [133](#)
  - serialize, [133](#)
  - size, [133](#)
- EMF::EMRSETWINDOWORGEX, [134](#)
  - EMRSETWINDOWORGEX, [134](#)
  - execute, [136](#)
  - serialize, [136](#)

- size, 136
- EMF::EMRSETWORLDTRANSFORM, 137
  - EMRSETWORLDTRANSFORM, 137
  - execute, 138
  - serialize, 138
  - size, 138
- EMF::EMRSTROKEANDFILLPATH, 139
  - EMRSTROKEANDFILLPATH, 139
  - execute, 140
  - serialize, 140
  - size, 140
- EMF::EMRSTROKEPATH, 141
  - EMRSTROKEPATH, 141
  - execute, 142
  - serialize, 142
  - size, 142
- EMF::ENHMETAHEADER, 143
  - ~ENHMETAHEADER, 143
  - ENHMETAHEADER, 143
  - execute, 144
  - serialize, 144
  - size, 144
  - unserialize, 144
- EMF::EXTPEN, 145
  - EXTPEN, 145
  - getType, 145
  - newEMR, 146
- EMF::FONT, 146
  - FONT, 147
  - getType, 147
  - newEMR, 147
- EMF::GLOBALOBJECTS, 148
  - add, 150
  - begin, 150
  - end, 151
  - find, 151
  - newRecord, 151
  - remove, 151
- EMF::GRAPHICSOBJECT, 153
  - contexts, 154
  - newEMR, 153
- EMF::INTARRAY, 154
  - INTARRAY, 154
- EMF::METAFILEDEVICECONTEXT, 155
  - ~METAFILEDEVICECONTEXT, 157
  - appendHandle, 157
  - appendRecord, 157
  - clearHandle, 157
  - deleteMetafile, 158
  - ds, 159
  - emf\_handles, 159
  - fp, 159
  - getType, 158
  - handles, 159
  - header, 159
  - mergePoint, 158
  - METAFILEDEVICECONTEXT, 156
  - nextHandle, 158
  - records, 159
- EMF::METARECORD, 160
  - ~METARECORD, 161
  - execute, 161
  - serialize, 161
  - size, 162
- EMF::OBJECT, 162
  - getType, 163
  - handle, 163
  - OBJECT, 163
- EMF::PADDING, 164
  - PADDING, 164
- EMF::PALETTE, 165
  - getType, 166
  - newEMR, 166
  - PALETTE, 165
- EMF::PEN, 166
  - getType, 167
  - newEMR, 167
  - PEN, 167
- EMF::POINT16ARRAY, 168
  - POINT16ARRAY, 168
- EMF::POINTLARRAY, 169
  - POINTLARRAY, 169
- EMF::WCHARSTR, 170
  - WCHARSTR, 170
- emf\_handles
  - EMF::METAFILEDEVICECONTEXT, 159
- EMRARC
  - EMF::EMRARC, 30
- EMRARCTO
  - EMF::EMRARCTO, 32
- EMRBEGINPATH
  - EMF::EMRBEGINPATH, 34
- EMRCLOSEFIGURE
  - EMF::EMRCLOSEFIGURE, 36
- EMRCREATEBRUSHINDIRECT
  - EMF::EMRCREATEBRUSHINDIRECT, 37, 38
- EMRCREATEPALETTE
  - EMF::EMRCREATEPALETTE, 39, 40
- EMRCREATEPEN
  - EMF::EMRCREATEPEN, 41, 42
- EMRDELETEOBJECT
  - EMF::EMRDELETEOBJECT, 43, 44
- EMRELLIPSE
  - EMF::EMRELLIPSE, 45, 46
- EMRENDPATH
  - EMF::EMRENDPATH, 47
- EMREOF
  - EMF::EMREOF, 49
- EMREXTCREATEFONTINDIRECTW
  - EMF::EMREXTCREATEFONTINDIRECTW, 51
- EMREXTCREATEPEN
  - EMF::EMREXTCREATEPEN, 53
- EMREXTTEXTOUTA
  - EMF::EMREXTTEXTOUTA, 56
- EMREXTTEXTOUTW
  - EMF::EMREXTTEXTOUTW, 58, 59

EMRFILLPATH  
  EMF::EMRFILLPATH, [60](#), [61](#)  
EMRLINETO  
  EMF::EMRLINETO, [62](#), [63](#)  
EMRMODIFYWORLDTRANSFORM  
  EMF::EMRMODIFYWORLDTRANSFORM, [64](#)  
EMRMOVETOEX  
  EMF::EMRMOVETOEX, [66](#)  
EMRPOLYBEZIER  
  EMF::EMRPOLYBEZIER, [68](#)  
EMRPOLYBEZIER16  
  EMF::EMRPOLYBEZIER16, [70](#), [71](#)  
EMRPOLYBEZIERTO  
  EMF::EMRPOLYBEZIERTO, [74](#)  
EMRPOLYBEZIERTO16  
  EMF::EMRPOLYBEZIERTO16, [76](#), [77](#)  
EMRPOLYGON  
  EMF::EMRPOLYGON, [78](#), [79](#)  
EMRPOLYGON16  
  EMF::EMRPOLYGON16, [80](#), [81](#)  
EMRPOLYLINE  
  EMF::EMRPOLYLINE, [84](#)  
EMRPOLYLINE16  
  EMF::EMRPOLYLINE16, [86](#), [87](#)  
EMRPOLYLINETO  
  EMF::EMRPOLYLINETO, [88](#), [89](#)  
EMRPOLYLINETO16  
  EMF::EMRPOLYLINETO16, [90](#), [91](#)  
EMRPOLYPOLYGON  
  EMF::EMRPOLYPOLYGON, [94](#)  
EMRPOLYPOLYGON16  
  EMF::EMRPOLYPOLYGON16, [96](#), [97](#)  
EMRRECTANGLE  
  EMF::EMRRECTANGLE, [98](#), [99](#)  
EMRRESTOREDC  
  EMF::EMRRESTOREDC, [100](#)  
EMRSAVEDC  
  EMF::EMRSAVEDC, [102](#)  
EMRSCALEVIEWPORTEXT  
  EMF::EMRSCALEVIEWPORTEXT, [104](#)  
EMRSCALEWINDOWEXT  
  EMF::EMRSCALEWINDOWEXT, [106](#)  
EMRSELECTOBJECT  
  EMF::EMRSELECTOBJECT, [108](#)  
EMRSETBKCOLOR  
  EMF::EMRSETBKCOLOR, [110](#)  
EMRSETBKMODE  
  EMF::EMRSETBKMODE, [112](#)  
EMRSETMAPMODE  
  EMF::EMRSETMAPMODE, [113](#), [114](#)  
EMRSETMETARGN  
  EMF::EMRSETMETARGN, [115](#)  
EMRSETMITERLIMIT  
  EMF::EMRSETMITERLIMIT, [117](#)  
EMRSETPIXELV  
  EMF::EMRSETPIXELV, [119](#)  
EMRSETPOLYFILLMODE  
  EMF::EMRSETPOLYFILLMODE, [122](#)  
EMRSETTEXTALIGN  
  EMF::EMRSETTEXTALIGN, [124](#)  
EMRSETTEXTCOLOR  
  EMF::EMRSETTEXTCOLOR, [126](#)  
EMRSETVIEWPORTEXT  
  EMF::EMRSETVIEWPORTEXT, [128](#)  
EMRSETVIEWPORTORGE  
  EMF::EMRSETVIEWPORTORGE, [130](#)  
EMRSETWINDOWEXT  
  EMF::EMRSETWINDOWEXT, [132](#)  
EMRSETWINDOWORGE  
  EMF::EMRSETWINDOWORGE, [134](#)  
EMRSETWORLDTRANSFORM  
  EMF::EMRSETWORLDTRANSFORM, [137](#)  
EMRSTROKEANDFILLPATH  
  EMF::EMRSTROKEANDFILLPATH, [139](#)  
EMRSTROKEPATH  
  EMF::EMRSTROKEPATH, [141](#)  
end  
  EMF::GLOBALOBJECTS, [151](#)  
ENHMETAHEADER  
  EMF::ENHMETAHEADER, [143](#)  
execute  
  EMF::EMRARC, [30](#)  
  EMF::EMRARCTO, [32](#)  
  EMF::EMRBEGINPATH, [34](#)  
  EMF::EMRCLOSEFIGURE, [36](#)  
  EMF::EMRCREATEBRUSHINDIRECT, [38](#)  
  EMF::EMRCREATEPALETTE, [40](#)  
  EMF::EMRCREATEPEN, [42](#)  
  EMF::EMRDELETEOBJECT, [44](#)  
  EMF::EMRELLIPSE, [46](#)  
  EMF::EMRENDPATH, [48](#)  
  EMF::EMREOF, [50](#)  
  EMF::EMREXTCREATEFONTINDIRECTW, [51](#)  
  EMF::EMREXTCREATEPEN, [53](#)  
  EMF::EMREXTTEXTOUTA, [57](#)  
  EMF::EMREXTTEXTOUTW, [59](#)  
  EMF::EMRFILLPATH, [61](#)  
  EMF::EMRLINETO, [63](#)  
  EMF::EMRMODIFYWORLDTRANSFORM, [65](#)  
  EMF::EMRMOVETOEX, [67](#)  
  EMF::EMRPOLYBEZIER, [69](#)  
  EMF::EMRPOLYBEZIER16, [71](#)  
  EMF::EMRPOLYBEZIERTO, [75](#)  
  EMF::EMRPOLYBEZIERTO16, [77](#)  
  EMF::EMRPOLYGON, [79](#)  
  EMF::EMRPOLYGON16, [81](#)  
  EMF::EMRPOLYLINE, [85](#)  
  EMF::EMRPOLYLINE16, [87](#)  
  EMF::EMRPOLYLINETO, [89](#)  
  EMF::EMRPOLYLINETO16, [91](#)  
  EMF::EMRPOLYPOLYGON, [95](#)  
  EMF::EMRPOLYPOLYGON16, [97](#)  
  EMF::EMRRECTANGLE, [99](#)  
  EMF::EMRRESTOREDC, [101](#)  
  EMF::EMRSAVEDC, [103](#)  
  EMF::EMRSCALEVIEWPORTEXT, [104](#)



- EMF::EMRSCALEWINDOWEXTTEX, 106
- EMF::EMRSELECTOBJECT, 108
- EMF::EMRSETBKCOLOR, 110
- EMF::EMRSETBKMODE, 112
- EMF::EMRSETMAPMODE, 114
- EMF::EMRSETMETARGN, 116
- EMF::EMRSETMITERLIMIT, 118
- EMF::EMRSETPIXELV, 121
- EMF::EMRSETPOLYFILLMODE, 123
- EMF::EMRSETTEXTALIGN, 125
- EMF::EMRSETTEXTCOLOR, 127
- EMF::EMRSETVIEWPORTEXTTEX, 129
- EMF::EMRSETVIEWPORTORGEX, 131
- EMF::EMRSETWINDOWEXTTEX, 133
- EMF::EMRSETWINDOWORGEX, 136
- EMF::EMRSETWORLDTRANSFORM, 138
- EMF::EMRSTROKEANDFILLPATH, 140
- EMF::EMRSTROKEPATH, 142
- EMF::ENHMETAHEADER, 144
- EMF::METARECORD, 161
- EXTPEN
  - EMF::EXTPEN, 145
- find
  - EMF::GLOBALOBJECTS, 151
- FONT
  - EMF::FONT, 147
- fp
  - EMF::METAFILEDEVICECONTEXT, 159
- getType
  - EMF::BRUSH, 8
  - EMF::EXTPEN, 145
  - EMF::FONT, 147
  - EMF::METAFILEDEVICECONTEXT, 158
  - EMF::OBJECT, 163
  - EMF::PALETTE, 166
  - EMF::PEN, 167
- handle
  - EMF::OBJECT, 163
- handles
  - EMF::METAFILEDEVICECONTEXT, 159
- header
  - EMF::METAFILEDEVICECONTEXT, 159
- INTARRAY
  - EMF::INTARRAY, 154
- mergePoint
  - EMF::METAFILEDEVICECONTEXT, 158
- METAFILEDEVICECONTEXT
  - EMF::METAFILEDEVICECONTEXT, 156
- newEMR
  - EMF::BRUSH, 8
  - EMF::EXTPEN, 146
  - EMF::FONT, 147
  - EMF::GRAPHICSOBJECT, 153
  - EMF::PALETTE, 166
  - EMF::PEN, 167
- newRecord
  - EMF::GLOBALOBJECTS, 151
- nextHandle
  - EMF::METAFILEDEVICECONTEXT, 158
- OBJECT
  - EMF::OBJECT, 163
- operator<<
  - EMF::DATASTREAM, 12–17, 19, 20
- operator>>
  - EMF::DATASTREAM, 20–27
- PADDING
  - EMF::PADDING, 164
- PALETTE
  - EMF::PALETTE, 165
- PEN
  - EMF::PEN, 167
- POINT16ARRAY
  - EMF::POINT16ARRAY, 168
- POINTLARRAY
  - EMF::POINTLARRAY, 169
- records
  - EMF::METAFILEDEVICECONTEXT, 159
- remove
  - EMF::GLOBALOBJECTS, 151
- serialize
  - EMF::EMRARC, 31
  - EMF::EMRARCTO, 33
  - EMF::EMRBEGINPATH, 35
  - EMF::EMRCLOSEFIGURE, 36
  - EMF::EMRCREATEBRUSHINDIRECT, 38
  - EMF::EMRCREATEPALETTE, 40
  - EMF::EMRCREATEPEN, 42
  - EMF::EMRDELETEOBJECT, 44
  - EMF::EMRELLIPSE, 46
  - EMF::EMRENDPATH, 48
  - EMF::EMREOF, 50
  - EMF::EMREXTCREATEFONTINDIRECTW, 52
  - EMF::EMREXTCREATEPEN, 55
  - EMF::EMREXTTEXTOUTA, 57
  - EMF::EMREXTTEXTOUTW, 59
  - EMF::EMRFILLPATH, 61
  - EMF::EMRLINETO, 63
  - EMF::EMRMODIFYWORLDTRANSFORM, 65
  - EMF::EMRMOVETOEX, 67
  - EMF::EMRPOLYBEZIER, 69
  - EMF::EMRPOLYBEZIER16, 73
  - EMF::EMRPOLYBEZIERTO, 75
  - EMF::EMRPOLYBEZIERTO16, 77
  - EMF::EMRPOLYGON, 79
  - EMF::EMRPOLYGON16, 83
  - EMF::EMRPOLYLINE, 85
  - EMF::EMRPOLYLINE16, 87
  - EMF::EMRPOLYLINETO, 89
  - EMF::EMRPOLYLINETO16, 93

- EMF::EMRPOLYPOLYGON, [95](#)
- EMF::EMRPOLYPOLYGON16, [97](#)
- EMF::EMRRECTANGLE, [99](#)
- EMF::EMRRESTOREDC, [101](#)
- EMF::EMRSAVEDC, [103](#)
- EMF::EMRSCALEVIEWPORTEXTEX, [105](#)
- EMF::EMRSCALEWINDOWEXTTEX, [107](#)
- EMF::EMRSELECTOBJECT, [109](#)
- EMF::EMRSETBKCOLOR, [111](#)
- EMF::EMRSETBKMODE, [112](#)
- EMF::EMRSETMAPMODE, [114](#)
- EMF::EMRSETMETARGN, [116](#)
- EMF::EMRSETMITERLIMIT, [118](#)
- EMF::EMRSETPIXELV, [121](#)
- EMF::EMRSETPOLYFILLMODE, [123](#)
- EMF::EMRSETTEXTALIGN, [125](#)
- EMF::EMRSETTEXTCOLOR, [127](#)
- EMF::EMRSETVIEWPORTEXTEX, [129](#)
- EMF::EMRSETVIEWPORTORGEX, [131](#)
- EMF::EMRSETWINDOWEXTTEX, [133](#)
- EMF::EMRSETWINDOWORGEX, [136](#)
- EMF::EMRSETWORLDTRANSFORM, [138](#)
- EMF::EMRSTROKEANDFILLPATH, [140](#)
- EMF::EMRSTROKEPATH, [142](#)
- EMF::ENHMETAHEADER, [144](#)
- EMF::METARECORD, [161](#)
- setStream
  - EMF::DATASTREAM, [28](#)
- size
  - EMF::EMRARC, [31](#)
  - EMF::EMRARCTO, [33](#)
  - EMF::EMRBEGINPATH, [35](#)
  - EMF::EMRCLOSEFIGURE, [37](#)
  - EMF::EMRCREATEBRUSHINDIRECT, [38](#)
  - EMF::EMRCREATEPALETTE, [40](#)
  - EMF::EMRCREATEPEN, [42](#)
  - EMF::EMRDELETEOBJECT, [44](#)
  - EMF::EMRELLIPSE, [46](#)
  - EMF::EMRENDPATH, [48](#)
  - EMF::EMREOF, [50](#)
  - EMF::EMREXTCREATEFONTINDIRECTW, [52](#)
  - EMF::EMREXTCREATEPEN, [55](#)
  - EMF::EMREXTTEXTOUTA, [57](#)
  - EMF::EMREXTTEXTOUTW, [59](#)
  - EMF::EMRFILLPATH, [61](#)
  - EMF::EMRLINETO, [63](#)
  - EMF::EMRMODIFYWORLDTRANSFORM, [65](#)
  - EMF::EMRMOVETOEX, [67](#)
  - EMF::EMRPOLYBEZIER, [69](#)
  - EMF::EMRPOLYBEZIER16, [73](#)
  - EMF::EMRPOLYBEZIERTO, [75](#)
  - EMF::EMRPOLYBEZIERTO16, [77](#)
  - EMF::EMRPOLYGON, [80](#)
  - EMF::EMRPOLYGON16, [83](#)
  - EMF::EMRPOLYLINE, [85](#)
  - EMF::EMRPOLYLINE16, [87](#)
  - EMF::EMRPOLYLINETO, [90](#)
  - EMF::EMRPOLYLINETO16, [93](#)
- EMF::EMRPOLYPOLYGON, [95](#)
- EMF::EMRPOLYPOLYGON16, [97](#)
- EMF::EMRRECTANGLE, [99](#)
- EMF::EMRRESTOREDC, [101](#)
- EMF::EMRSAVEDC, [103](#)
- EMF::EMRSCALEVIEWPORTEXTEX, [105](#)
- EMF::EMRSCALEWINDOWEXTTEX, [107](#)
- EMF::EMRSELECTOBJECT, [109](#)
- EMF::EMRSETBKCOLOR, [111](#)
- EMF::EMRSETBKMODE, [113](#)
- EMF::EMRSETMAPMODE, [114](#)
- EMF::EMRSETMETARGN, [116](#)
- EMF::EMRSETMITERLIMIT, [118](#)
- EMF::EMRSETPIXELV, [121](#)
- EMF::EMRSETPOLYFILLMODE, [123](#)
- EMF::EMRSETTEXTALIGN, [125](#)
- EMF::EMRSETTEXTCOLOR, [127](#)
- EMF::EMRSETVIEWPORTEXTEX, [129](#)
- EMF::EMRSETVIEWPORTORGEX, [131](#)
- EMF::EMRSETWINDOWEXTTEX, [133](#)
- EMF::EMRSETWINDOWORGEX, [136](#)
- EMF::EMRSETWORLDTRANSFORM, [138](#)
- EMF::EMRSTROKEANDFILLPATH, [140](#)
- EMF::EMRSTROKEPATH, [142](#)
- EMF::ENHMETAHEADER, [144](#)
- EMF::METARECORD, [162](#)
- unserialize
  - EMF::ENHMETAHEADER, [144](#)
- WCHARSTR
  - EMF::WCHARSTR, [170](#)