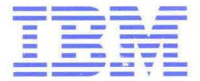


*Presentation Manager  
Programming Reference Vol II*



**OS/2<sup>®</sup> WARP**

**Version 3**

*Presentation Manager  
Programming Reference Vol II*



**OS/2<sup>®</sup> WARP**

**Version 3**

## Note

Before using this information and the product it supports, be sure to read the general information under Appendix I, "Notices" on page I-1.

### First Edition (October 1994)

**The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM authorized reseller or IBM marketing representative.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood NY 10594, U.S.A.

**COPYRIGHT LICENSE:** This publication contains printed sample application programs in source language, which illustrate OS/2 programming techniques. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the OS/2 application programming interface.

Each copy of any portion of these sample programs or any derivative work, which is distributed to others, must include a copyright notice as follows: "© (your company name) (year). All rights reserved."

© Copyright International Business Machines Corporation 1994. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

|  |       |
|--|-------|
| <b>Figures</b> .....   | xxv   |
| <b>Chapter 9. Introduction to Message Processing</b> .....           | 9-1   |
| Message Types .....  | 9-1   |
| Default Window and Dialog Procedure Message Processing .....         | 9-2   |
| Control Window Message Processing .....                              | 9-2   |
| Notation Conventions .....   | 9-4   |
| <b>Chapter 10. Default Window Procedure Message Processing</b> ..... | 10-1  |
| Purpose .....  | 10-1  |
| Reserved Messages .....  | 10-1  |
| General Window Styles .....  | 10-1  |
| Window Class Styles .....  | 10-2  |
| Window Styles .....  | 10-3  |
| General Window Messages .....  | 10-5  |
| PL_ALTERED .....   | 10-5  |
| WM_ACTIVATE .....  | 10-5  |
| WM_APPTERMINATENOTIFY .....  | 10-7  |
| WM_ADJUSTWINDOWPOS .....   | 10-7  |
| WM_BEGINDRAG .....   | 10-9  |
| WM_BEGINSELECT .....   | 10-10 |
| WM_BUTTON1CLICK .....  | 10-11 |
| WM_BUTTON1DBLCLK .....   | 10-12 |
| WM_BUTTON1DOWN .....   | 10-13 |
| WM_BUTTON1MOTIONEND .....  | 10-14 |
| WM_BUTTON1MOTIONSTART .....  | 10-15 |
| WM_BUTTON1UP .....   | 10-16 |
| WM_BUTTON2CLICK .....  | 10-17 |
| WM_BUTTON2DBLCLK .....   | 10-18 |
| WM_BUTTON2DOWN .....   | 10-19 |
| WM_BUTTON2MOTIONEND .....  | 10-20 |
| WM_BUTTON2MOTIONSTART .....  | 10-21 |
| WM_BUTTON2UP .....   | 10-22 |
| WM_BUTTON3CLICK .....  | 10-23 |
| WM_BUTTON3DBLCLK .....   | 10-24 |
| WM_BUTTON3DOWN .....   | 10-25 |
| WM_BUTTON3MOTIONEND .....  | 10-26 |
| WM_BUTTON3MOTIONSTART .....  | 10-27 |
| WM_BUTTON3UP .....   | 10-28 |
| WM_CALCFRAMERECT .....   | 10-29 |
| WM_CALCVALIDRECTS .....  | 10-30 |
| WM_CHAR .....  | 10-32 |
| WM_CHORD .....   | 10-35 |
| WM_CLOSE .....   | 10-35 |
| WM_COMMAND .....   | 10-37 |

|                      |       |
|----------------------|-------|
| WM_CONTEXTMENU       | 10-38 |
| WM_CONTROL           | 10-39 |
| WM_CONTROLPOINTER    | 10-40 |
| WM_CREATE            | 10-41 |
| WM_DESTROY           | 10-42 |
| WM_DRAWITEM          | 10-42 |
| WM_ENABLE            | 10-43 |
| WM_ENDDRAG           | 10-44 |
| WM_ENDSELECT         | 10-45 |
| WM_ERROR             | 10-46 |
| WM_FOCUSCHANGE       | 10-47 |
| WM_FORMATFRAME       | 10-48 |
| WM_HELP              | 10-49 |
| WM_HITTEST           | 10-50 |
| WM_HSCROLL           | 10-51 |
| WM_INITDLG           | 10-52 |
| WM_INITMENU          | 10-53 |
| WM_JOURNALNOTIFY     | 10-54 |
| WM_MATCHMNEMONIC     | 10-55 |
| WM_MEASUREITEM       | 10-55 |
| WM_MENUEND           | 10-56 |
| WM_MENUSELECT        | 10-57 |
| WM_MINMAXFRAME       | 10-58 |
| WM_MOUSEMAP          | 10-59 |
| WM_MOUSEMOVE         | 10-59 |
| WM_MOVE              | 10-60 |
| WM_MSGBOXDISMISS     | 10-62 |
| WM_MSGBOXINIT        | 10-62 |
| WM_NEXTMENU          | 10-63 |
| WM_NULL              | 10-64 |
| WM_OPEN              | 10-65 |
| WM_PACTIVATE         | 10-65 |
| WM_PAINT             | 10-66 |
| WM_PCONTROL          | 10-67 |
| WM_PPAIN             | 10-68 |
| WM_PRESPARAMCHANGED  | 10-69 |
| WM_PSETFOCUS         | 10-69 |
| WM_PSIZE             | 10-70 |
| WM_PSYSCOLORCHANGE   | 10-71 |
| WM_QUERYACCELTABLE   | 10-72 |
| WM_QUERYCONVERTPOS   | 10-72 |
| WM_QUERYHELPINFO     | 10-74 |
| WM_QUERYTRACKINFO    | 10-74 |
| WM_QUERYWINDOWPARAMS | 10-75 |
| WM_QUIT              | 10-76 |
| WM_REALIZEPALETTE    | 10-78 |
| WM_SAVEAPPLICATION   | 10-78 |
| WM_SEM1              | 10-79 |

|                                       |        |
|---------------------------------------|--------|
| WM_SEM2                               | 10-80  |
| WM_SEM3                               | 10-81  |
| WM_SEM4                               | 10-82  |
| WM_SETACCELTABLE                      | 10-83  |
| WM_SETFOCUS                           | 10-83  |
| WM_SETHelpINFO                        | 10-84  |
| WM_SETSELECTION                       | 10-85  |
| WM_SETWINDOWPARAMS                    | 10-86  |
| WM_SHOW                               | 10-87  |
| WM_SINGLESELECT                       | 10-87  |
| WM_SIZE                               | 10-88  |
| WM_SUBSTITUTESTRING                   | 10-89  |
| WM_SYSCOLORCHANGE                     | 10-90  |
| WM_SYSCOMMAND                         | 10-91  |
| WM_SYSVALUECHANGED                    | 10-92  |
| WM_TEXTEDIT                           | 10-93  |
| WM_TIMER                              | 10-94  |
| WM_TRACKFRAME                         | 10-95  |
| WM_TRANSLATEACCEL                     | 10-95  |
| WM_TRANSLATEMNEMONIC                  | 10-96  |
| WM_UPDATEFRAME                        | 10-97  |
| WM_VRNDISABLED                        | 10-98  |
| WM_VRNENABLED                         | 10-98  |
| WM_VSCROLL                            | 10-99  |
| WM_WINDOWPOSCHANGED                   | 10-101 |
| Default Dialog Processing             | 10-103 |
| WM_CHAR (Default Dialogs)             | 10-103 |
| WM_CLOSE (Default Dialogs)            | 10-104 |
| WM_COMMAND (Default Dialogs)          | 10-104 |
| WM_INITDLG (Default Dialogs)          | 10-104 |
| WM_MATCHMNEMONIC (Default Dialogs)    | 10-105 |
| WM_QUERYDLGCODE                       | 10-105 |
| Default File Dialog Processing        | 10-106 |
| FDM_ERROR                             | 10-107 |
| FDM_FILTER                            | 10-108 |
| FDM_VALIDATE                          | 10-108 |
| Default Font Dialog Processing        | 10-109 |
| WM_DRAWITEM (in Font Dialog)          | 10-110 |
| FNTM_FACENAMECHANGED                  | 10-111 |
| FNTM_FILTERLIST                       | 10-112 |
| FNTM_POINTSIZACHANGED                 | 10-113 |
| FNTM_STYLECHANGED                     | 10-114 |
| FNTM_UPDATEPREVIEW                    | 10-115 |
| Language Support Window Processing    | 10-116 |
| WM_ACTIVATE (Language Support Window) | 10-116 |
| WM_CONTROL (Language Support Window)  | 10-116 |
| WM_PAINT (Language Support Window)    | 10-117 |
| WM_PPAINT (Language Support Window)   | 10-117 |

|   |        |
|---|--------|
| WM_SETFOCUS (Language Support Window)       | 10-118 |
| WM_SIZE (Language Support Window)           | 10-118 |
| WM_SYSCOLORCHANGE (Language Support Window) | 10-119 |
| Language Support Dialog Processing          | 10-120 |
| WM_ACTIVATE (Language Support Dialog)       | 10-120 |
| WM_CONTROL (Language Support Dialog)        | 10-120 |
| WM_PAINT (Language Support Dialog)          | 10-121 |
| WM_PPAINT (Language Support Dialog)         | 10-121 |
| WM_SETFOCUS (Language Support Dialog)       | 10-122 |
| WM_SIZE (Language Support Dialog)           | 10-122 |
| WM_SYSCOLORCHANGE (Language Support Dialog) | 10-122 |

|   |       |
|---|-------|
| <b>Chapter 11. Button Control Window Processing</b> | 11-1  |
| Purpose   | 11-1  |
| Button Control Styles                               | 11-1  |
| Button Control Data                                 | 11-3  |
| Default Colors                                      | 11-3  |
| Button Control Notification Messages                | 11-5  |
| WM_COMMAND (in Button Controls)                     | 11-5  |
| WM_CONTROL (in Button Controls)                     | 11-5  |
| WM_HELP (in Button Controls)                        | 11-7  |
| WM_SYSCOMMAND                                       | 11-7  |
| Button Control Window Messages                      | 11-8  |
| BM_CLICK  | 11-8  |
| BM_QUERYCHECK                                       | 11-9  |
| BM_QUERYCHECKINDEX                                  | 11-9  |
| BM_QUERYHILITE                                      | 11-10 |
| BM_SETCHECK   | 11-11 |
| BM_SETDEFAULT                                       | 11-12 |
| BM_SETHILITE  | 11-13 |
| WM_ENABLE (in Button Controls)                      | 11-14 |
| WM_MATCHMNEMONIC (in Button Controls)               | 11-14 |
| WM_QUERYCONVERTPOS (in Button Controls)             | 11-15 |
| WM_QUERYWINDOWPARAMS (in Button Controls)           | 11-15 |
| WM_SETWINDOWPARAMS (in Button Controls)             | 11-15 |

|  |      |
|--|------|
| <b>Chapter 12. Entry Field Control Window Processing</b> | 12-1 |
| Purpose  | 12-1 |
| Entry Field Control Styles                               | 12-1 |
| Entry Field Control Data                                 | 12-3 |
| Default Colors   | 12-3 |
| Entry Field Control Notification Messages                | 12-4 |
| WM_CONTROL (in Entry Fields)                             | 12-4 |
| Entry Field Control Window Messages                      | 12-6 |
| EM_CLEAR   | 12-6 |
| EM_COPY  | 12-6 |
| EM_CUT   | 12-7 |
| EM_PASTE   | 12-8 |

|  |             |
|--|-------------|
| EM_QUERYCHANGED                                    | 12-9        |
| EM_QUERYFIRSTCHAR                                  | 12-9        |
| EM_QUERYREADONLY                                   | 12-10       |
| EM_QUERYSEL  | 12-11       |
| EM_SETFIRSTCHAR                                    | 12-12       |
| EM_SETINSERTMODE                                   | 12-13       |
| EM_SETREADONLY                                     | 12-13       |
| EM_SETSEL  | 12-14       |
| EM_SETTEXTLIMIT                                    | 12-15       |
| WM_CHAR (in Entry Fields)                          | 12-16       |
| WM_QUERYCONVERTPOS (in Entry Fields)               | 12-17       |
| WM_QUERYWINDOWPARAMS (in Entry Fields)             | 12-17       |
| WM_SETWINDOWPARAMS (in Entry Fields)               | 12-18       |
| <b>Chapter 13. Frame Control Window Processing</b> | <b>13-1</b> |
| Purpose  | 13-1        |
| Frame Creation Flags                               | 13-1        |
| Frame Control Styles                               | 13-3        |
| Frame Control Data                                 | 13-3        |
| Default Colors                                     | 13-4        |
| Frame Control Notification Messages                | 13-5        |
| WM_MINMAXFRAME (in Frame Controls)                 | 13-5        |
| Frame Control Window Messages                      | 13-8        |
| WM_ACTIVATE (in Frame Controls)                    | 13-8        |
| WM_ADJUSTFRAMEPOS                                  | 13-8        |
| WM_BUTTON1DBLCLK (in Frame Controls)               | 13-10       |
| WM_BUTTON2DBLCLK (in Frame Controls)               | 13-10       |
| WM_BUTTON1DOWN (in Frame Controls)                 | 13-10       |
| WM_BUTTON2DOWN (in Frame Controls)                 | 13-11       |
| WM_BUTTON1UP (in Frame Controls)                   | 13-11       |
| WM_BUTTON2UP (in Frame Controls)                   | 13-11       |
| WM_CALCFRAMERECT (in Frame Controls)               | 13-12       |
| WM_CHAR (in Frame Controls)                        | 13-12       |
| WM_CLOSE (in Frame Controls)                       | 13-13       |
| WM_COMMAND   | 13-13       |
| WM_DRAWITEM (in Frame Controls)                    | 13-13       |
| WM_ERASEBACKGROUND                                 | 13-14       |
| WM_FLASHWINDOW                                     | 13-15       |
| WM_FOCUSCHANGE (in Frame Controls)                 | 13-16       |
| WM_FORMATFRAME (in Frame Controls)                 | 13-16       |
| WM_INITMENU (in Frame Controls)                    | 13-17       |
| WM_MEASUREITEM (in Frame Controls)                 | 13-17       |
| WM_MENUSELECT (in Frame Controls)                  | 13-18       |
| WM_NEXTMENU (in Frame Controls)                    | 13-18       |
| WM_OWNERPOSCHANGE                                  | 13-18       |
| WM_PAINT (in Frame Controls)                       | 13-19       |
| WM_QUERYBORDERSIZE                                 | 13-20       |
| WM_QUERYCONVERTPOS (in Frame Controls)             | 13-20       |



|   |             |
|---|-------------|
| WM_QUERYFOCUSCHAIN . . . . .                                    | 13-21       |
| WM_QUERYFRAMECTLCOUNT . . . . .                                 | 13-22       |
| WM_QUERYFRAMEINFO . . . . .                                     | 13-23       |
| WM_QUERYICON . . . . .  | 13-24       |
| WM_QUERYWINDOWPARAMS (in Frame Controls) . . . . .              | 13-24       |
| WM_SETBORDERSIZE . . . . .                                      | 13-25       |
| WM_SETICON . . . . .  | 13-25       |
| WM_SETWINDOWPARAMS (in Frame Controls) . . . . .                | 13-26       |
| WM_SIZE (in Frame Controls) . . . . .                           | 13-26       |
| WM_SYSCOMMAND . . . . .   | 13-27       |
| WM_TRACKFRAME (in Frame Controls) . . . . .                     | 13-29       |
| WM_TRANSLATEACCEL (in Frame Controls) . . . . .                 | 13-30       |
| WM_TRANSLATEMNEMONIC (in Frame Controls) . . . . .              | 13-30       |
| WM_UPDATEFRAME (in Frame Controls) . . . . .                    | 13-30       |
| <br>  |             |
| <b>Chapter 14. List Box Control Window Processing . . . . .</b> | <b>14-1</b> |
| Purpose . . . . .   | 14-1        |
| List Box Control Styles . . . . .                               | 14-1        |
| List Box Control Data . . . . .                                 | 14-1        |
| Default Colors . . . . .  | 14-1        |
| List Box Control Notification Messages . . . . .                | 14-3        |
| WM_CONTROL (in List Boxes) . . . . .                            | 14-3        |
| WM_DRAWITEM (in List Boxes) . . . . .                           | 14-4        |
| WM_MEASUREITEM (in List Boxes) . . . . .                        | 14-5        |
| List Box Control Window Messages . . . . .                      | 14-7        |
| LM_DELETEALL . . . . .  | 14-7        |
| LM_DELETEITEM . . . . .   | 14-7        |
| LM_INSERTITEM . . . . .   | 14-8        |
| LM_INSERTMULTITEMS . . . . .                                    | 14-9        |
| LM_QUERYITEMCOUNT . . . . .                                     | 14-10       |
| LM_QUERYITEMHANDLE . . . . .                                    | 14-11       |
| LM_QUERYITEMTEXT . . . . .                                      | 14-12       |
| LM_QUERYITEMTEXTLENGTH . . . . .                                | 14-13       |
| LM_QUERYSELECTION . . . . .                                     | 14-13       |
| LM_QUERYTOPINDEX . . . . .                                      | 14-15       |
| LM_SEARCHSTRING . . . . .                                       | 14-15       |
| LM_SELECTITEM . . . . .   | 14-17       |
| LM_SETITEMHANDLE . . . . .                                      | 14-18       |
| LM_SETITEMHEIGHT . . . . .                                      | 14-19       |
| LM_SETITEMTEXT . . . . .  | 14-19       |
| LM_SETITEMWIDTH . . . . .                                       | 14-20       |
| LM_SETTOPINDEX . . . . .  | 14-21       |
| WM_CHAR (in List Boxes) . . . . .                               | 14-22       |
| WM_QUERYCONVERTPOS (in List Boxes) . . . . .                    | 14-23       |
| WM_QUERYWINDOWPARAMS (in List Boxes) . . . . .                  | 14-23       |
| WM_SETWINDOWPARAMS (in List Boxes) . . . . .                    | 14-23       |
| <br>  |             |
| <b>Chapter 15. Menu Control Window Processing . . . . .</b>     | <b>15-1</b> |

|   |       |
|---|-------|
| Purpose   | 15-1  |
| Menu Control Styles   | 15-1  |
| Menu Item Styles  | 15-2  |
| Menu Item Attributes  | 15-3  |
| Default Colors  | 15-3  |
| Menu Control Notification Messages                                  | 15-5  |
| WM_COMMAND (in Menu Controls)                                       | 15-5  |
| WM_DRAWITEM (in Menu Controls)                                      | 15-5  |
| WM_HELP (in Menu Controls)  | 15-6  |
| WM_INITMENU (in Menu Controls)                                      | 15-7  |
| WM_MEASUREITEM (in Menu Controls)                                   | 15-7  |
| WM_MENUEND (in Menu Controls)                                       | 15-8  |
| WM_MENUSELECT (in Menu Controls)                                    | 15-8  |
| WM_NEXTMENU (in Menu Controls)                                      | 15-9  |
| Menu Control Window Messages  | 15-10 |
| MM_DELETEITEM   | 15-10 |
| MM_ENDMENUMODE  | 15-11 |
| MM_INSERTITEM   | 15-11 |
| MM_ISITEMVALID  | 15-12 |
| MM_ITEMIDFROMPOSITION   | 15-13 |
| MM_ITEMPOSITIONFROMID   | 15-14 |
| MM_QUERYDEFAULTITEMID   | 15-15 |
| MM_QUERYITEM  | 15-16 |
| MM_QUERYITEMATTR  | 15-17 |
| MM_QUERYITEMCOUNT   | 15-18 |
| MM_QUERYITEMRECT  | 15-18 |
| MM_QUERYITEMTEXT  | 15-19 |
| MM_QUERYITEMTEXTLENGTH  | 15-20 |
| MM_QUERYSELITEMID   | 15-21 |
| MM_REMOVEITEM   | 15-22 |
| MM_SELECTITEM   | 15-23 |
| MM_SETDEFAULTITEMID   | 15-24 |
| MM_SETITEM  | 15-25 |
| MM_SETITEMATTR  | 15-26 |
| MM_SETITEMHANDLE  | 15-27 |
| MM_SETITEMTEXT  | 15-28 |
| MM_STARTMENUMODE  | 15-29 |
| WM_QUERYCONVERTPOS (in Menu Controls)                               | 15-30 |
| WM_QUERYWINDOWPARAMS (in Menu Controls)                             | 15-30 |
| WM_SETWINDOWPARAMS (in Menu Controls)                               | 15-30 |
| WM_SYSCOMMAND   | 15-31 |
| <br>  |       |
| <b>Chapter 16. Multi-Line Entry Field Control Window Processing</b> | 16-1  |
| Purpose   | 16-1  |
| How to Use  | 16-1  |
| Multi-Line Entry Field Control Styles                               | 16-2  |
| Multi-Line Entry Field Control Data                                 | 16-2  |
| Multi-Line Entry Field Control Notification Messages                | 16-3  |

|  |       |
|--|-------|
| WM_CONTROL (in Multiline Entry Fields) | 16-3  |
| Multi-Line Entry Field Window Messages | 16-9  |
| MLM_CHARFROMLINE                       | 16-9  |
| MLM_CLEAR                              | 16-9  |
| MLM_COPY                               | 16-10 |
| MLM_CUT                                | 16-11 |
| MLM_DELETE                             | 16-12 |
| MLM_DISABLEREFRESH                     | 16-12 |
| MLM_ENABLEREFRESH                      | 16-13 |
| MLM_EXPORT                             | 16-14 |
| MLM_FORMAT                             | 16-15 |
| MLM_IMPORT                             | 16-16 |
| MLM_INSERT                             | 16-17 |
| MLM_LINEFROMCHAR                       | 16-18 |
| MLM_PASTE                              | 16-18 |
| MLM_QUERYBACKCOLOR                     | 16-19 |
| MLM_QUERYCHANGED                       | 16-20 |
| MLM_QUERYFIRSTCHAR                     | 16-20 |
| MLM_QUERYFONT                          | 16-21 |
| MLM_QUERYFORMATLINELENGTH              | 16-22 |
| MLM_QUERYFORMATRECT                    | 16-22 |
| MLM_QUERYFORMATTEXTLENGTH              | 16-23 |
| MLM_QUERYIMPORTEXPORT                  | 16-24 |
| MLM_QUERYLINECOUNT                     | 16-25 |
| MLM_QUERYLINELENGTH                    | 16-25 |
| MLM_QUERYREADONLY                      | 16-26 |
| MLM_QUERYSEL                           | 16-27 |
| MLM_QUERYSELTEXT                       | 16-29 |
| MLM_QUERYTABSTOP                       | 16-29 |
| MLM_QUERYTEXTCOLOR                     | 16-30 |
| MLM_QUERYTEXTLENGTH                    | 16-31 |
| MLM_QUERYTEXTLIMIT                     | 16-31 |
| MLM_QUERYUNDO                          | 16-32 |
| MLM_QUERYWRAP                          | 16-33 |
| MLM_RESETUNDO                          | 16-33 |
| MLM_SEARCH                             | 16-35 |
| MLM_SETBACKCOLOR                       | 16-37 |
| MLM_SETCHANGED                         | 16-38 |
| MLM_SETFIRSTCHAR                       | 16-38 |
| MLM_SETFONT                            | 16-39 |
| MLM_SETFORMATRECT                      | 16-40 |
| MLM_SETREADONLY                        | 16-43 |
| MLM_SETIMPORTEXPORT                    | 16-43 |
| MLM_SETSEL                             | 16-45 |
| MLM_SETTABSTOP                         | 16-46 |
| MLM_SETTEXTCOLOR                       | 16-46 |
| MLM_SETTEXTLIMIT                       | 16-47 |
| MLM_SETWRAP                            | 16-48 |

|  |       |
|--|-------|
| MLM_UNDO   | 16-49 |
| WM_BUTTON1DBLCLK (in Multiline Entry Fields)                 | 16-50 |
| WM_BUTTON1DOWN (in Multiline Entry Fields)                   | 16-51 |
| WM_BUTTON1UP (in Multiline Entry Fields)                     | 16-51 |
| WM_CHAR (in Multiline Entry Fields)                          | 16-52 |
| WM_ENABLE (in Multiline Entry Fields)                        | 16-55 |
| WM_MOUSEMOVE (in Multiline Entry Fields)                     | 16-56 |
| WM_QUERYWINDOWPARAMS (in Multiline Entry Fields)             | 16-56 |
| WM_SETWINDOWPARAMS (in Multiline Entry Fields)               | 16-57 |
| <b>Chapter 17. Combination-Box Control Window Processing</b> | 17-1  |
| Purpose  | 17-1  |
| Combination Box Control Styles                               | 17-1  |
| Combination Box Control Data                                 | 17-1  |
| Default Colors   | 17-2  |
| Combo Box Control Notification Messages                      | 17-3  |
| WM_CONTROL (in Combination Boxes)                            | 17-3  |
| Combo Box Control Window Messages                            | 17-5  |
| CBM_HILITE   | 17-6  |
| CBM_ISLISTSHOWING  | 17-7  |
| CBM_SHOWLIST   | 17-7  |
| <b>Chapter 18. Scroll Bar Control Window Processing</b>      | 18-1  |
| Purpose  | 18-1  |
| Scroll Bar Control Styles                                    | 18-1  |
| Scroll Bar Control Data                                      | 18-1  |
| Default Colors   | 18-1  |
| Scroll Bar System Values                                     | 18-2  |
| Scroll Bar Control Notification Messages                     | 18-3  |
| WM_HSCROLL (in Horizontal Scroll Bars)                       | 18-3  |
| WM_VSCROLL (in Vertical Scroll Bars)                         | 18-3  |
| Scroll Bar Control Window Messages                           | 18-4  |
| SBM_QUERYPOS   | 18-4  |
| SBM_QUERYRANGE   | 18-4  |
| SBM_SETPOS   | 18-5  |
| SBM_SETSCROLLBAR   | 18-6  |
| SBM_SETTHUMBSIZE   | 18-7  |
| WM_QUERYCONVERTPOS (in Scroll Bars)                          | 18-8  |
| WM_QUERYWINDOWPARAMS (in Scroll Bars)                        | 18-8  |
| WM_SETWINDOWPARAMS (in Scroll Bars)                          | 18-9  |
| <b>Chapter 19. Spin Button Control Window Processing</b>     | 19-1  |
| Purpose  | 19-1  |
| Spin Button Control Styles                                   | 19-1  |
| Spin Button Control Data                                     | 19-2  |
| Spin Button Control Notification Message                     | 19-3  |
| WM_CONTROL (in Spin Button Controls)                         | 19-3  |
| Spin Button Control Window Messages                          | 19-4  |

|  |             |
|--|-------------|
| SPBM_OVERRIDESETLIMITS                                 | 19-4        |
| SPBM_QUERYLIMITS                                       | 19-5        |
| SPBM_QUERYVALUE  | 19-6        |
| SPBM_SETARRAY  | 19-8        |
| SPBM_SETCURRENTVALUE                                   | 19-9        |
| SPBM_SETLIMITS   | 19-9        |
| SPBM_SETMASTER   | 19-10       |
| SPBM_SETTEXTLIMIT                                      | 19-11       |
| SPBM_SPINDOWN  | 19-12       |
| SPBM_SPINUP  | 19-13       |
| <b>Chapter 20. Static Control Window Processing</b>    | <b>20-1</b> |
| Purpose  | 20-1        |
| Static Control Styles                                  | 20-1        |
| Static Control Data                                    | 20-2        |
| Default Colors   | 20-2        |
| Static Control Notification Messages                   | 20-3        |
| Static Control Window Messages                         | 20-4        |
| SM_QUERYHANDLE   | 20-4        |
| SM_SETHANDLE   | 20-5        |
| WM_MATCHMnemonic (in Static Controls)                  | 20-5        |
| WM_QUERYCONVERTPOS (in Static Controls)                | 20-6        |
| WM_QUERYWINDOWPARAMS (in Static Controls)              | 20-6        |
| WM_SETWINDOWPARAMS (in Static Controls)                | 20-7        |
| <b>Chapter 21. Title Bar Control Window Processing</b> | <b>21-1</b> |
| Purpose  | 21-1        |
| Title Bar Control Styles                               | 21-1        |
| Title Bar Control Data                                 | 21-1        |
| Default Colors   | 21-1        |
| Title Bar Control Notification Messages                | 21-2        |
| WM_SYSCOMMAND (in Title Bar Controls)                  | 21-2        |
| WM_TRACKFRAME (in Title Bar Controls)                  | 21-2        |
| Title Bar Control Window Messages                      | 21-3        |
| TBM_QUERYHILITE  | 21-3        |
| TBM_SETHILITE  | 21-4        |
| WM_QUERYCONVERTPOS (in Title Bar Controls)             | 21-4        |
| WM_QUERYWINDOWPARAMS (in Title Bars)                   | 21-5        |
| WM_SETWINDOWPARAMS (in Title Bar Controls)             | 21-5        |
| <b>Chapter 22. Container Control Window Processing</b> | <b>22-1</b> |
| Purpose  | 22-1        |
| Container Control Window Words                         | 22-2        |
| Container Control Styles and Selection Types           | 22-2        |
| Container Control Styles                               | 22-2        |
| Container Control Selection Types                      | 22-3        |
| Container Control Data                                 | 22-4        |
| Container Control Notification Messages                | 22-5        |

|   |       |
|---|-------|
| WM_CONTROL (in Container Controls)        | 22-5  |
| WM_CONTROLPOINTER (in Container Controls) | 22-6  |
| WM_DRAWITEM (in Container Controls)       | 22-7  |
| Container Control Notification Codes      | 22-10 |
| CN_BEGINEDIT                              | 22-10 |
| CN_COLLAPSETREE                           | 22-11 |
| CN_CONTEXTMENU                            | 22-12 |
| CN_DRAGAFTER                              | 22-12 |
| CN_DRAGLEAVE                              | 22-15 |
| CN_DRAGOVER                               | 22-16 |
| CN_DROP                                   | 22-18 |
| CN_DROPNOTIFY                             | 22-19 |
| CN_DROPHELP                               | 22-19 |
| CN_EMPHASIS                               | 22-20 |
| CN_ENDEDIT                                | 22-21 |
| CN_ENTER                                  | 22-22 |
| CN_EXPANDTREE                             | 22-23 |
| CN_HELP                                   | 22-23 |
| CN_INITDRAG                               | 22-24 |
| CN_KILLFOCUS                              | 22-25 |
| CN_PICKUP                                 | 22-26 |
| CN_QUERYDELTA                             | 22-27 |
| CN_REALLOCPSZ                             | 22-28 |
| CN_SCROLL                                 | 22-29 |
| CN_SETFOCUS                               | 22-29 |
| Container Control Window Messages         | 22-31 |
| CM_ALLOCDETAILFIELDINFO                   | 22-31 |
| CM_ALLOCRECORD                            | 22-32 |
| CM_ARRANGE                                | 22-34 |
| CM_CLOSEEDIT                              | 22-35 |
| CM_COLLAPSETREE                           | 22-36 |
| CM_ERASERECORD                            | 22-37 |
| CM_EXPANDTREE                             | 22-38 |
| CM_FILTER                                 | 22-39 |
| CM_FREEDetailFIELDINFO                    | 22-40 |
| CM_FREERECORD                             | 22-42 |
| CM_HORZSCROLLSPLITWINDOW                  | 22-43 |
| CM_INSERTDETAILFIELDINFO                  | 22-44 |
| CM_INSERTRECORD                           | 22-45 |
| CM_INSERTRECORDARRAY                      | 22-47 |
| CM_INVALIDATEDetailFIELDINFO              | 22-49 |
| CM_INVALIDATERECORD                       | 22-50 |
| CM_MOVETREE                               | 22-52 |
| CM_OPENEDIT                               | 22-54 |
| CM_PAINTBACKGROUND                        | 22-55 |
| CM_QUERYCNRINFO                           | 22-56 |
| CM_QUERYDETAILFIELDINFO                   | 22-56 |
| CM_QUERYDRAGIMAGE                         | 22-57 |

|   |             |
|---|-------------|
| CM_QUERYRECORD  | 22-59       |
| CM_QUERYRECORDEMPHASIS                                | 22-60       |
| CM_QUERYRECORDFROMRECT                                | 22-62       |
| CM_QUERYRECORDINFO                                    | 22-63       |
| CM_QUERYRECORDRECT                                    | 22-64       |
| CM_QUERYVIEWPORTRECT                                  | 22-65       |
| CM_REMOVEDetailFieldInfo                              | 22-66       |
| CM_REMOVERECORD                                       | 22-68       |
| CM_SCROLLWINDOW                                       | 22-70       |
| CM_SEARCHSTRING                                       | 22-71       |
| CM_SETCNINFO  | 22-72       |
| CM_SETRECORDEMPHASIS                                  | 22-74       |
| CM_SORTRECORD   | 22-76       |
| CM_SETTEXTVISIBILITY                                  | 22-77       |
| WM_PICKUP   | 22-78       |
| WM_PRESPARAMCHANGED (in Container Controls)           | 22-79       |
| <br>  |             |
| <b>Chapter 23. Notebook Control Window Processing</b> | <b>23-1</b> |
| Purpose   | 23-1        |
| Notebook Control Styles                               | 23-1        |
| Notebook Control Data                                 | 23-2        |
| Notebook Control Notification Messages                | 23-3        |
| WM_CONTROL (in Notebook Controls)                     | 23-3        |
| WM_CONTROLPOINTER (in Notebook Controls)              | 23-4        |
| WM_DRAWITEM (in Notebook Controls)                    | 23-4        |
| Notebook Control Window Messages                      | 23-7        |
| BKM_CALCPAGERECT                                      | 23-7        |
| BKM_DELETEPAGE  | 23-8        |
| BKM_INSERTPAGE  | 23-9        |
| BKM_INVALIDATETABS                                    | 23-11       |
| BKM_QUERYPAGECOUNT                                    | 23-11       |
| BKM_QUERYPAGEDATA                                     | 23-13       |
| BKM_QUERYPAGEID                                       | 23-13       |
| BKM_QUERYPAGEINFO                                     | 23-15       |
| BKM_QUERYPAGESTYLE                                    | 23-16       |
| BKM_QUERYPAGEWINDOWHWND                               | 23-17       |
| BKM_QUERYSTATUSLINETEXT                               | 23-18       |
| BKM_QUERYTABBITMAP                                    | 23-19       |
| BKM_QUERYTABTEXT                                      | 23-20       |
| BKM_SETDIMENSIONS                                     | 23-21       |
| BKM_SETNOTEBOOKCOLORS                                 | 23-22       |
| BKM_SETPAGEDATA                                       | 23-23       |
| BKM_SETPAGEINFO                                       | 23-24       |
| BKM_SETPAGEWINDOWHWND                                 | 23-25       |
| BKM_SETSTATUSLINETEXT                                 | 23-26       |
| BKM_SETTABBITMAP                                      | 23-26       |
| BKM_SETTABTEXT  | 23-28       |
| BKM_TURNTOPAGE  | 23-29       |

|  |       |
|--|-------|
| WM_CHAR (in Notebook Controls)                             | 23-30 |
| WM_PRESPARAMCHANGED (in Notebook Controls)                 | 23-31 |
| WM_SIZE (in Notebook Controls)                             | 23-32 |
| <b>Chapter 24. Slider Control Window Processing</b>        | 24-1  |
| Purpose  | 24-1  |
| Slider Control Styles                                      | 24-1  |
| Slider Control Data  | 24-4  |
| Slider Control Notification Messages                       | 24-5  |
| WM_CONTROL (in Slider Controls)                            | 24-5  |
| WM_CONTROLPOINTER (in Slider Controls)                     | 24-6  |
| WM_DRAWITEM (in Slider Controls)                           | 24-6  |
| Slider Control Window Messages                             | 24-8  |
| SLM_ADDDETENT  | 24-8  |
| SLM_QUERYDETENTPOS   | 24-9  |
| SLM_QUERYSCALETEXT   | 24-10 |
| SLM_QUERYSLIDERINFO  | 24-11 |
| SLM_QUERYTICKPOS   | 24-13 |
| SLM_QUERYTICKSIZE  | 24-14 |
| SLM_REMOVEDETENT   | 24-15 |
| SLM_SETSCALETEXT   | 24-16 |
| SLM_SETSLIDERINFO  | 24-17 |
| SLM_SETTICKSIZE  | 24-19 |
| WM_CHAR (in Slider Controls)                               | 24-20 |
| WM_PRESPARAMCHANGED (in Slider Controls)                   | 24-22 |
| WM_QUERYWINDOWPARAMS (in Slider Controls)                  | 24-23 |
| WM_SETWINDOWPARAMS (in Slider Controls)                    | 24-24 |
| <b>Chapter 25. Circular Slider Control Window Messages</b> | 25-1  |
| Purpose  | 25-1  |
| Circular Slider Control Styles                             | 25-1  |
| Circular Slider Control Data                               | 25-2  |
| Default Colors   | 25-2  |
| Circular Slider Control Notification Messages              | 25-3  |
| WM_CONTROL (in Circular Slider Controls)                   | 25-3  |
| WM_CONTROLPOINTER (in Circular Slider Controls)            | 25-4  |
| Circular Slider Control Window Messages                    | 25-5  |
| CSM_QUERYINCREMENT   | 25-5  |
| CSM_QUERYRADIUS  | 25-5  |
| CSM_QUERYRANGE   | 25-6  |
| CSM_QUERYVALUE   | 25-6  |
| CSM_SETBITMAPDATA  | 25-7  |
| CSM_SETINCREMENT   | 25-7  |
| CSM_SETRANGE   | 25-8  |
| CSM_SETVALUE   | 25-9  |
| WM_CHAR (in Circular Slider Controls)                      | 25-9  |
| WM_PRESPARAMCHANGED (in Circular Slider Controls)          | 25-10 |
| WM_QUERYWINDOWPARAMS (in Circular Slider Controls)         | 25-11 |



|  |       |
|--|-------|
| WM_SETWINDOWPARAMS (in Circular Slider Controls)       | 25-12 |
| <b>Chapter 26. Value Set Control Window Processing</b> | 26-1  |
| Purpose  | 26-1  |
| Value Set Control Styles                               | 26-1  |
| Value Set Control Data                                 | 26-5  |
| Value Set Control Notification Messages                | 26-6  |
| WM_CONTROL (in Value Set Controls)                     | 26-6  |
| WM_CONTROLPOINTER (in Value Set Controls)              | 26-7  |
| WM_DRAWITEM (in Value Set Controls)                    | 26-8  |
| Value Set Control Window Messages                      | 26-10 |
| VM_QUERYITEM   | 26-10 |
| VM_QUERYITEMATTR                                       | 26-12 |
| VM_QUERYMETRICS  | 26-14 |
| VM_QUERYSELECTEDITEM                                   | 26-15 |
| VM_SELECTITEM  | 26-16 |
| VM_SETITEM   | 26-17 |
| VM_SETITEMATTR   | 26-19 |
| VM_SETMETRICS  | 26-21 |
| WM_CHAR (in Value Set Controls)                        | 26-22 |
| WM_PRESPARAMCHANGED (in Value Set Controls)            | 26-24 |
| WM_QUERYWINDOWPARAMS (in Value Set Controls)           | 26-25 |
| WM_SETWINDOWPARAMS (in Value Set Controls)             | 26-26 |
| WM_SIZE (in Value Set Controls)                        | 26-26 |
| <b>Chapter 27. Clipboard Messages</b>                  | 27-1  |
| Purpose  | 27-1  |
| WM_DESTROYCLIPBOARD                                    | 27-1  |
| WM_DRAWCLIPBOARD                                       | 27-2  |
| WM_HSCROLLCLIPBOARD                                    | 27-2  |
| WM_PAINTCLIPBOARD                                      | 27-4  |
| WM_RENDERALLFMTS                                       | 27-4  |
| WM_RENDERFMT   | 27-5  |
| WM_SIZECLIPBOARD                                       | 27-6  |
| WM_VSCROLLCLIPBOARD                                    | 27-7  |
| <b>Chapter 28. Direct Manipulation (Drag) Messages</b> | 28-1  |
| Purpose  | 28-1  |
| DM_DISCARDOBJECT                                       | 28-1  |
| DM_DRAGERROR   | 28-2  |
| DM_DRAGFILECOMPLETE                                    | 28-3  |
| DM_DRAGLEAVE   | 28-4  |
| DM_DRAGOVER  | 28-4  |
| DM_DRAGOVERNOTIFY                                      | 28-7  |
| DM_DROP  | 28-8  |
| DM_DROPHELP  | 28-9  |
| DM_DROPNOTIFY  | 28-10 |
| DM_EMPHASIZETARGET                                     | 28-11 |

|   |             |
|---|-------------|
| DM_ENDCONVERSATION . . . . .                                | 28-11       |
| DM_FILERENDERED . . . . .                                   | 28-12       |
| DM_PRINTOBJECT . . . . .                                    | 28-13       |
| DM_RENDER . . . . .   | 28-14       |
| DM_RENDERCOMPLETE . . . . .                                 | 28-15       |
| DM_RENDERFILE . . . . .                                     | 28-17       |
| DM_RENDERPREPARE . . . . .                                  | 28-18       |
| <br>  |             |
| <b>Chapter 29. Dynamic Data Exchange Messages . . . . .</b> | <b>29-1</b> |
| Purpose . . . . .   | 29-1        |
| WM_DDE_ACK . . . . .  | 29-1        |
| WM_DDE_ADVISE . . . . .                                     | 29-2        |
| WM_DDE_DATA . . . . .                                       | 29-3        |
| WM_DDE_EXECUTE . . . . .                                    | 29-4        |
| WM_DDE_INITIATE . . . . .                                   | 29-4        |
| WM_DDE_INITIATEACK . . . . .                                | 29-6        |
| WM_DDE_POKE . . . . .                                       | 29-6        |
| WM_DDE_REQUEST . . . . .                                    | 29-7        |
| WM_DDE_TERMINATE . . . . .                                  | 29-8        |
| WM_DDE_UNADVISE . . . . .                                   | 29-9        |
| <br>  |             |
| <b>Chapter 30. Help Manager Messages . . . . .</b>          | <b>30-1</b> |
| Purpose . . . . .   | 30-1        |
| HM_ACTIONBAR_COMMAND . . . . .                              | 30-1        |
| HM_CONTROL . . . . .  | 30-1        |
| HM_CREATE_HELP_TABLE . . . . .                              | 30-2        |
| HM_DISMISS_WINDOW . . . . .                                 | 30-3        |
| HM_DISPLAY_HELP . . . . .                                   | 30-4        |
| HM_ERROR . . . . .  | 30-5        |
| HM_EXT_HELP . . . . .                                       | 30-7        |
| HM_EXT_HELP_UNDEFINED . . . . .                             | 30-8        |
| HM_GENERAL_HELP . . . . .                                   | 30-8        |
| HM_GENERAL_HELP_UNDEFINED . . . . .                         | 30-9        |
| HM_HELP_CONTENTS . . . . .                                  | 30-10       |
| HM_HELP_INDEX . . . . .                                     | 30-10       |
| HM_HELPSUBITEM_NOT_FOUND . . . . .                          | 30-11       |
| HM_INFORM . . . . .   | 30-12       |
| HM_INVALIDATE_DDF_DATA . . . . .                            | 30-13       |
| HM_KEYS_HELP . . . . .                                      | 30-14       |
| HM_LOAD_HELP_TABLE . . . . .                                | 30-15       |
| HM_NOTIFY . . . . .   | 30-15       |
| HM_QUERY . . . . .  | 30-17       |
| HM_QUERY_DDF_DATA . . . . .                                 | 30-19       |
| HM_QUERY_KEYS_HELP . . . . .                                | 30-20       |
| HM_REPLACE_HELP_FOR_HELP . . . . .                          | 30-20       |
| HM_REPLACE_USING_HELP . . . . .                             | 30-21       |
| HM_SET_ACTIVE_WINDOW . . . . .                              | 30-22       |
| HM_SET_COVERPAGE_SIZE . . . . .                             | 30-23       |

|  |             |
|--|-------------|
| HM_SET_HELP_LIBRARY_NAME                             | 30-24       |
| HM_SET_HELP_WINDOW_TITLE                             | 30-25       |
| HM_SET_OBJCOM_WINDOW                                 | 30-25       |
| HM_SET_SHOW_PANEL_ID                                 | 30-26       |
| HM_SET_USERDATA                                      | 30-27       |
| HM_TUTORIAL  | 30-27       |
| HM_UPDATE_OBJCOM_WINDOW_CHAIN                        | 30-28       |
| <b>Chapter 31. Resource Files</b>                    | <b>31-1</b> |
| How to Read the Syntax Definitions                   | 31-1        |
| Definitions Used in all Resources                    | 31-2        |
| Specification of Values                              | 31-2        |
| Resource Load and Memory Options                     | 31-2        |
| Resource Script File Specification                   | 31-2        |
| Single-Line Statements                               | 31-3        |
| User-Defined Resources                               | 31-4        |
| RCDATA statement                                     | 31-5        |
| Directives   | 31-6        |
| Multiple-Line Statements                             | 31-9        |
| Keyboard Resources                                   | 31-10       |
| ACCELTABLE Statement                                 | 31-10       |
| ASSOCTABLE Statement                                 | 31-12       |
| Dialog and Window Template Statements                | 31-13       |
| MENU Statement                                       | 31-16       |
| STRINGTABLE Statement                                | 31-22       |
| Templates, Control Data, and Presentation Parameters | 31-24       |
| Dialog Template                                      | 31-24       |
| Dialog Coordinates                                   | 31-24       |
| Dialog Template Format and Contents                  | 31-24       |
| Header   | 31-26       |
| Items  | 31-26       |
| Data Area  | 31-27       |
| Control Data Statement                               | 31-28       |
| Presentation Parameters Statement                    | 31-28       |
| Parent/Child/Owner Relationship                      | 31-29       |
| Predefined Window Classes                            | 31-30       |
| Predefined Control Statements                        | 31-30       |
| Resource (.RES) File Specification                   | 31-35       |
| <b>Chapter 32. Code Pages</b>                        | <b>32-1</b> |
| Windowed PM Applications                             | 32-1        |
| OS/2 Code Page Options for PM Applications           | 32-3        |
| OS/2 Font Support for Multiple Code Pages            | 32-4        |
| Font Code-Page Functions                             | 32-4        |
| Font Layout  | 32-4        |
| ASCII Code Pages                                     | 32-12       |
| EBCDIC Code Pages                                    | 32-21       |

|                               |      |
|-------------------------------|------|
| <b>Appendix A. Data Types</b> | A-1  |
| ACCEL                         | A-1  |
| ACCELTABLE                    | A-1  |
| APIRET                        | A-2  |
| APSZ                          | A-2  |
| ARCPARAMS                     | A-3  |
| AREABUNDLE                    | A-3  |
| ATOM                          | A-4  |
| BITMAPARRAYFILEHEADER         | A-4  |
| BITMAPARRAYFILEHEADER2        | A-5  |
| BITMAPFILEHEADER              | A-6  |
| BITMAPFILEHEADER2             | A-7  |
| BITMAPINFO                    | A-8  |
| BITMAPINFO2                   | A-9  |
| BITMAPINFOHEADER              | A-14 |
| BITMAPINFOHEADER2             | A-15 |
| BIT16                         | A-20 |
| BIT32                         | A-20 |
| BIT8                          | A-20 |
| BOOL                          | A-21 |
| BOOKPAGEINFO                  | A-21 |
| BOOKTEXT                      | A-23 |
| BTNCDATA                      | A-24 |
| BYTE                          | A-24 |
| CATCHBUF                      | A-25 |
| CDATE                         | A-25 |
| CHAR                          | A-26 |
| CHARBUNDLE                    | A-26 |
| CLASSINFO                     | A-27 |
| CNRDRAGINFO                   | A-28 |
| CNRDRAWITEMINFO               | A-28 |
| CNREDITDATA                   | A-29 |
| CNRDRAGINIT                   | A-32 |
| CNRINFO                       | A-33 |
| CNRLAZYDRAGINFO               | A-39 |
| COLOR                         | A-40 |
| CONVCONTEXT                   | A-40 |
| CREATESTRUCT                  | A-41 |
| CSBITMAPDATA                  | A-42 |
| CURSORINFO                    | A-43 |
| CTIME                         | A-44 |
| Control-Data                  | A-44 |
| DDEINIT                       | A-45 |
| DELETENOTIFY                  | A-46 |
| DDESTRUCT                     | A-46 |
| DESKTOP                       | A-48 |
| DEVOPENSTRUC                  | A-49 |
| DLGTEMPLATE                   | A-53 |

|                 |       |
|-----------------|-------|
| DLGTITEM        | A-54  |
| DRAGIMAGE       | A-56  |
| DRAGINFO        | A-57  |
| DRAGITEM        | A-58  |
| DRAGTRANSFER    | A-61  |
| DRIVDATA        | A-63  |
| ENTRYFDATA      | A-64  |
| ERRORID         | A-65  |
| ERRINFO         | A-65  |
| ESCMODE         | A-66  |
| ESCSETMODE      | A-67  |
| FACENAMEDESC    | A-68  |
| FATTRS          | A-69  |
| FFDESCS         | A-72  |
| FIELDINFO       | A-72  |
| FIELDINFOINSERT | A-75  |
| FILEDLG         | A-77  |
| FIXED           | A-81  |
| FONTDLG         | A-82  |
| FONTMETRICS     | A-88  |
| FRAMECDATA      | A-99  |
| GRADIENTL       | A-100 |
| HAB             | A-100 |
| HACCEL          | A-101 |
| HAPP            | A-101 |
| HATOMTBL        | A-101 |
| HBITMAP         | A-101 |
| HDC             | A-102 |
| HCINFO          | A-102 |
| HDDF            | A-103 |
| HELPINIT        | A-104 |
| HELPSUBTABLE    | A-106 |
| HELPTABLE       | A-108 |
| HENUM           | A-108 |
| HEV             | A-108 |
| HINI            | A-109 |
| HLIB            | A-109 |
| HMF             | A-109 |
| HMODULE         | A-109 |
| HMQ             | A-110 |
| HMTX            | A-110 |
| HMUX            | A-110 |
| HOBJECT         | A-110 |
| HPOINTER        | A-111 |
| HPROGRAM        | A-111 |
| HPS             | A-111 |
| HRGN            | A-111 |
| HSAVEWP         | A-112 |

|                      |       |
|----------------------|-------|
| HSEM                 | A-112 |
| HSPL                 | A-112 |
| HSTR                 | A-112 |
| HSWITCH              | A-113 |
| HWND                 | A-113 |
| ICONINFO             | A-113 |
| IMAGEBUNDLE          | A-114 |
| IPT                  | A-115 |
| KERNINGPAIRS         | A-115 |
| LBOXINFO             | A-116 |
| LHANDLE              | A-117 |
| LINEBUNDLE           | A-117 |
| LONG                 | A-118 |
| MARKERBUNDLE         | A-119 |
| MATRIXLF             | A-120 |
| MB2D                 | A-121 |
| MB2INFO              | A-121 |
| MENUITEM             | A-123 |
| MINIRECORDCORE       | A-124 |
| MLE_SEARCHDATA       | A-125 |
| MLEMARGSTRUCT        | A-126 |
| MLECTLDATA           | A-127 |
| MPARAM               | A-129 |
| MQINFO               | A-129 |
| MRESULT              | A-130 |
| NOTIFYDELTA          | A-130 |
| NOTIFYRECORDEMPHASIS | A-131 |
| NOTIFYRECORDENTER    | A-132 |
| NOTIFYSCROLL         | A-133 |
| OBJCLASS             | A-134 |
| OWNERBACKGROUND      | A-135 |
| OWNERITEM            | A-136 |
| MLEOVERFLOW          | A-137 |
| PAGEINFO             | A-138 |
| PAGESELECTNOTIFY     | A-140 |
| PANOSE               | A-140 |
| PARAM                | A-144 |
| PCH                  | A-147 |
| PCSZ                 | A-147 |
| PDEVOPENDATA         | A-147 |
| PFN                  | A-148 |
| PFNWP                | A-148 |
| PID                  | A-148 |
| PIX                  | A-149 |
| PRDINFO3             | A-149 |
| PRDRIVINFO           | A-151 |
| PRESPARAMS           | A-151 |
| PRINTDEST            | A-152 |

|                  |       |
|------------------|-------|
| PRINTERINFO      | A-153 |
| PRFPROFILE       | A-154 |
| PRJINFO2         | A-155 |
| PRJINFO3         | A-157 |
| PROGRAMENTRY     | A-159 |
| PROGCATEGORY     | A-160 |
| PROGDETAILS      | A-160 |
| PROGTYPE         | A-161 |
| PRPORTINFO       | A-162 |
| PRPORTINFO1      | A-163 |
| PRQINFO3         | A-164 |
| PRQINFO6         | A-166 |
| PRQPROCINFO      | A-169 |
| POINTERINFO      | A-169 |
| POINTL           | A-170 |
| POINTS           | A-171 |
| PQMOPENDATA      | A-171 |
| PSZ              | A-171 |
| PWPOINT          | A-172 |
| PVOID            | A-172 |
| QMSG             | A-172 |
| QUERYRECFROMRECT | A-173 |
| QUERYRECORDRECT  | A-174 |
| RECORDCORE       | A-175 |
| RECORDINSERT     | A-177 |
| RECTL            | A-179 |
| RENDERFILE       | A-179 |
| RGB              | A-180 |
| RGB2             | A-181 |
| RGNRECT          | A-182 |
| SBCDATA          | A-182 |
| SEARCHSTRING     | A-184 |
| SEGOFF           | A-185 |
| SFACTORS         | A-185 |
| SHORT            | A-186 |
| SIZEF            | A-186 |
| SIZEL            | A-186 |
| SLDCDATA         | A-187 |
| SMHSTRUCT        | A-188 |
| SPBCDATA         | A-189 |
| SPLERR           | A-190 |
| STR16            | A-190 |
| STR32            | A-190 |
| STR64            | A-190 |
| STR8             | A-191 |
| STYLECHANGE      | A-191 |
| SWBLOCK          | A-193 |
| SWCNTRL          | A-193 |

|  |            |
|--|------------|
| SWENTRY  | A-195      |
| SWP  | A-195      |
| TID  | A-197      |
| TRACKINFO  | A-197      |
| TREEITEMDESC   | A-199      |
| TREEMOVE   | A-200      |
| UCHAR  | A-201      |
| ULONG  | A-201      |
| USERBUTTON   | A-202      |
| USHORT   | A-202      |
| VIOSIZECOUNT   | A-203      |
| VIOFONTCELLSIZE  | A-203      |
| VOID   | A-204      |
| VSCDATA  | A-204      |
| VSDRAGINFO   | A-205      |
| VSDRAGINIT   | A-205      |
| VSTEXT   | A-206      |
| WNDPARAMS  | A-207      |
| WPOINT   | A-208      |
| WRECT  | A-208      |
| XYWINSIZE  | A-208      |
| <b>Appendix B. Error Codes</b>                                   | <b>B-1</b> |
| <b>Appendix C. Error Explanations</b>                            | <b>C-1</b> |
| <b>Appendix D. Standard Bit-Map Formats</b>                      | <b>D-1</b> |
| Bit-Map Data   | D-1        |
| Bit-Map Information Tables                                       | D-1        |
| Bit-Map Example  | D-2        |
| Bit-Map File Format  | D-2        |
| <b>Appendix E. Fonts Supplied with the OS/2 Operating System</b> | <b>E-1</b> |
| OS/2 Outline Fonts   | E-1        |
| Presentation Manager Bit Map Fonts                               | E-2        |
| Fonts Supplied for ISO 9241 Non-Conforming Hardware              | E-2        |
| Fonts Supplied for ISO 9241 Conforming Hardware                  | E-5        |
| International Standards Organization (ISO) 9241                  | E-7        |
| <b>Appendix F. Format of Interchange Files</b>                   | <b>F-1</b> |
| Metafile Restrictions  | F-1        |
| Metafile Data Format   | F-3        |
| Structured Field Formats   | F-4        |
| <b>Appendix G. Initialization File Information</b>               | <b>G-1</b> |
| <b>Appendix H. Virtual Key Definitions</b>                       | <b>H-1</b> |



|                                      |      |
|--------------------------------------|------|
| <b>Appendix I. Notices</b> . . . . . | I-1  |
| Trademarks . . . . .                 | I-1  |
| <b>Glossary</b> . . . . .            | X-1  |
| Glossary Listing . . . . .           | X-1  |
| <b>Index</b> . . . . .               | X-29 |

## Figures

|        |   |       |
|--------|---|-------|
| 23-1.  | Tabs Showing Rectangular Area Used to Size a Bit Map . . . . .                | 23-27 |
| 26-1.  | Value Set with Bit Maps . . . . .   | 26-2  |
| 26-2.  | Value Set with Icons . . . . .  | 26-2  |
| 26-3.  | Value Set with Text Strings . . . . .   | 26-3  |
| 26-4.  | Value Set with Colors . . . . .   | 26-3  |
| 26-5.  | Value Set with Border . . . . .   | 26-4  |
| 26-6.  | Value Set with Item Borders . . . . .   | 26-4  |
| 31-1.  | Dialog Template . . . . .   | 31-25 |
| 32-1.  | OS/2 Code Page Options for PM Applications . . . . .                          | 32-3  |
| 32-2.  | US-English: ASCII Code Page 437 . . . . .                                     | 32-12 |
| 32-3.  | Latin 1 Multilingual: ASCII Code Page 850 . . . . .                           | 32-13 |
| 32-4.  | Latin 2 Multilingual: ASCII Code Page 852 . . . . .                           | 32-14 |
| 32-5.  | Turkey: ASCII Code Page 857 . . . . .   | 32-15 |
| 32-6.  | Portuguese: ASCII Code Page 860 . . . . .                                     | 32-16 |
| 32-7.  | Iceland: ASCII Code Page 861 . . . . .  | 32-17 |
| 32-8.  | Canadian-French: ASCII Code Page 863 . . . . .                                | 32-18 |
| 32-9.  | Norwegian: ASCII Code Page 865 . . . . .                                      | 32-19 |
| 32-10. | Desktop Publishing: ASCII Code Page 1004 . . . . .                            | 32-20 |
| 32-11. | US-English: EBCDIC Code Page 037 . . . . .                                    | 32-21 |
| 32-12. | Austrian/German: EBCDIC Code Page 273 . . . . .                               | 32-22 |
| 32-13. | Belgian: EBCDIC Code Page 274 (supported for migration purposes) . . . . .    | 32-23 |
| 32-14. | Danish/Norwegian: EBCDIC Code Page 277 . . . . .                              | 32-24 |
| 32-15. | Finnish/Swedish: EBCDIC Code Page 278 . . . . .                               | 32-25 |
| 32-16. | Italian: EBCDIC Code Page 280 . . . . .                                       | 32-26 |
| 32-17. | Portuguese: EBCDIC Code Page 282 (supported for migration purposes) . . . . . | 32-27 |
| 32-18. | Spanish: EBCDIC Code Page 284 . . . . .                                       | 32-28 |
| 32-19. | UK-English: EBCDIC Code Page 285 . . . . .                                    | 32-29 |
| 32-20. | French: EBCDIC Code Page 297 . . . . .  | 32-30 |
| 32-21. | International: EBCDIC Code Page 500 . . . . .                                 | 32-31 |
| 32-22. | Czechoslovakia/Hungary/Poland/Yugoslavia: EBCDIC Code Page 870 . . . . .      | 32-32 |
| 32-23. | Iceland: EBCDIC Code Page 871 . . . . .                                       | 32-33 |
| 32-24. | Turkey: EBCDIC Code Page 1026 . . . . .                                       | 32-34 |



---

## Chapter 9. Introduction to Message Processing

Messages are processed by window and dialog procedures.

Every window has a window procedure. Windows can also be combined into standard windows or dialog boxes. These are special cases of groups of windows that also have their own procedures. A window or dialog procedure must be capable of processing any message. This can be achieved by delegating some message types to the default window, or dialog, procedures by use of the `WinDefWindowProc` and `WinDefDlgProc` functions respectively.

Control windows are a special type of child windows. They take the form of objects such as buttons, scroll bars, list boxes, and text entry fields. These child windows process mouse and keyboard input and notify its owner of significant input events. Procedures for these child window controls are inside the Presentation Manager and are often called system-provided window procedures.

All messages have the same form as `QMSG` structure, which has the following form:

```
typedef struct _QMSG {
    HWND      hwnd;
    ULONG     msg;
    MPARAM    mp1;
    MPARAM    mp2;
    ULONG     time;
    POINTL    pt1;
    ULONG     reserved;
} QMSG;

typedef QMSG *PQMSG;
```

---

### Message Types

There are two types of window procedure message processing:

- Default window and dialog procedure message processing
- Control window message processing.

These types are described below along with the notation conventions used in the message descriptions. The messages are described in the following chapters.

## Default Window and Dialog Procedure Message Processing

These window procedures provide default processing for application window procedures:

- Default window and dialog procedure
- Language support window and dialog procedures, which are used if the application specifies a null window procedure
- Default AVIO window procedure.

These messages are described in Chapter 10, "Default Window Procedure Message Processing" on page 10-1. The system-provided window procedures take no action on messages that are not defined in this chapter, and return NULL.

## Control Window Message Processing

Controls are predefined classes of child windows that any application can use for input and output. These control classes are predefined:

|                          |   |
|--------------------------|---|
| <b>WC_BUTTON</b>         | Consists of buttons and boxes that the operator can select by clicking the pointing device or using the keyboard. These messages are described in Chapter 11, "Button Control Window Processing" on page 11-1.  |
| <b>WC_CIRCULARSLIDER</b> | Consists of a visual component whose specific purpose is to allow a user to set, display, or modify a value by moving the slider arm around the circular slider dial. Messages are described in Chapter 25, "Circular Slider Control Window Messages" on page 25-1.   |
| <b>WC_COMBOBOX</b>       | Consists of an entry field control and a list box control merged into a single control. The list, which is usually limited in size, is displayed below the entry field and offset one dialog box unit to its right. These messages are described in Chapter 17, "Combination-Box Control Window Processing" on page 17-1. |
| <b>WC_CONTAINER</b>      | Consists of a visual component whose specific purpose is to hold objects such as executable programs, word processing files, graphics images, and database records. Messages are described in Chapter 22, "Container Control Window Processing" on page 22-1.   |
| <b>WC_ENTRYFIELD</b>     | Consists of a single line of text that the operator can edit. These messages are described in Chapter 12, "Entry Field Control Window Processing" on page 12-1.   |
| <b>WC_FRAME</b>          | Consists of a composite window. These messages are described in Chapter 13, "Frame Control Window Processing" on page 13-1.   |
| <b>WC_LISTBOX</b>        | Presents a list of text items from which the operator can make selections. These messages are described in Chapter 14, "List Box Control Window Processing" on page 14-1.   |

|                      |  |
|----------------------|--|
| <b>WC_MENU</b>       | Presents a list of items, which may be text displayed horizontally as action bars or vertically as pull-down menus. Menus are usually used to provide a command interface to applications. These messages are described in Chapter 15, "Menu Control Window Processing" on page 15-1.  |
| <b>WC_MLE</b>        | Consists of a rectangular window that displays multiple lines of text that the operator can edit. When it has the focus, the cursor marks the current <i>insertion</i> or <i>replacement</i> point. These messages are described in Chapter 16, "Multi-Line Entry Field Control Window Processing" on page 16-1.   |
| <b>WC_NOTEBOOK</b>   | Consists of a visual component whose specific purpose is to organize information on individual pages so that a user can find and display that information quickly and easily. Messages are described in Chapter 23, "Notebook Control Window Processing" on page 23-1.   |
| <b>WC_SCROLLBAR</b>  | Consists of window scroll bars that allow the operator to make a request to scroll the contents of an associated window. These messages are described in Chapter 18, "Scroll Bar Control Window Processing" on page 18-1.  |
| <b>WC_SLIDER</b>     | Consists of a visual component whose specific purpose is to allow a user to set, display, or modify a value by moving the slider arm along the slider shaft. Messages are described in Chapter 24, "Slider Control Window Processing" on page 24-1.  |
| <b>WC_SPINBUTTON</b> | Presents a scrollable ring of choices from which the operator can select. These messages are described in Chapter 19, "Spin Button Control Window Processing" on page 19-1.  |
| <b>WC_STATIC</b>     | Consists of simple display items that do not respond to keyboard or pointing device events. These messages are described in Chapter 20, "Static Control Window Processing" on page 20-1.   |
| <b>WC_TITLEBAR</b>   | Displays the window title or caption and allows the operator to move its owner. These messages are described in Chapter 21, "Title Bar Control Window Processing" on page 21-1.  |
| <b>WC_VALUESET</b>   | Consists of a visual component whose specific purpose is to allow a user to select one choice from a group of mutually exclusive choices. A value set can use graphical images (bit maps or icons), as well as colors, text, and numbers, to represent the items that a user can select. Messages are described in Chapter 26, "Value Set Control Window Processing" on page 26-1. |

**Owner-Notification Messages:** Controls are useful because they notify their owners when significant events take place. A control notifies its owner by sending a WM\_CONTROL message or by posting a WM\_COMMAND or WM\_HELP message.

- WM\_CONTROL
- WM\_COMMAND

*Param2* contains information that indicates the source of the WM\_COMMAND message:

|                    |                                |
|--------------------|--------------------------------|
| CMDSRC_PUSHBUTTON  | Posted by a pushbutton control |
| CMDSRC_MENU        | Posted by a menu control       |
| CMDSRC_ACCELERATOR | Posted by WinTranslateAccel    |
| CMDSRC_FONTDLG     | Posted by a font dialog.       |
| CMDSRC_OTHER       | Other source.                  |

- WM\_HELP

*Param1* contains information that indicates the source of the WM\_HELP message:

|                    |                                |
|--------------------|--------------------------------|
| CMDSRC_PUSHBUTTON  | Posted by a pushbutton control |
| CMDSRC_MENU        | Posted by a menu control       |
| CMDSRC_ACCELERATOR | Posted by WinTranslateAccel    |
| CMDSRC_OTHER       | Other source.                  |

---

## Notation Conventions

Each message description contains:

- Name** The message name; a 2-byte identity unique to a message.
- Some message identity values are reserved for the use of the operating system, some are available for use by an application. See “Reserved Messages” on page 10-1.
- For all messages, the first two or three characters of the name indicate the type of window that is related to the message; for example:
- LM List box control  
SBM Scroll bar control.
- Cause** The principal reason that caused the generation of the message.
- Parameters** Input and output parameters pertinent to the message.
- There are always two parameters (*param1* and *param2*) and one *return* value. Any or all of the parameters can be NULL.
- Remarks** An explanation of the relationship between the parameters in the context of the message and an indication of the expected processing of the message.
- Default** A definition of how the default window procedures (provided by the system) process the message.
- Note:** A message is not equivalent to a call of the same name.

---

## Chapter 10. Default Window Procedure Message Processing

This system-provided window procedure processes the actions that control the operation of windows.

### Purpose

General window messages are used for standard processing. These messages can be requested from the system or sent to the system for information, or for actions such as create window, validate window, track mouse movement, and select and deselect actions.

---

### Reserved Messages

These message ranges are reserved:

**WM\_USER** All messages below this value are reserved for system use. Private messages must have an identifier with a value of WM\_USER or higher.

**Note:** The operating system uses certain message values higher than WM\_USER. These message values should not be used by an application. A partial listing of these messages is in the following figure:

From PMSTDDL.G.H:

```
#define FDM_FILTER           WM_USER+40
#define FDM_VALIDATE        WM_USER+41
#define FDM_ERROR           WM_USER+42

#define FNTM_FACENAMECHANGED WM_USER+50
#define FNTM_POINTSIZEDCHANGED WM_USER+51
#define FNTM_STYLECHANGED   WM_USER+52
#define FNTM_COLORCHANGED   WM_USER+53
#define FNTM_UPDATEPREVIEW  WM_USER+54
#define FNTM_FILTERLIST     WM_USER+55
```

You should scan your header files to see if other messages have been defined with values higher than WM\_USER.

---

### General Window Styles

The *window* is the mechanism by which the application communicates with the operator. Each window can have a window *style* that controls the appearance and behavior of the window. There are also *class* styles that apply to all the windows of a particular class (class being FRAME, BUTTON, and so on).



## Window Class Styles

These window class styles are available:

|                        |   |
|------------------------|---|
| <b>CS_SIZEREDRAW</b>   | Determines whether a window will be redrawn when sized. This style is to be used for a window whose contents are sensitive to the size of the window. For example, the data in some windows can be scaled up or down to fit the size of the Client Area. In other windows, the data remains the same size whatever the size of the window; it is merely clipped if the window is made smaller. The CS_SIZEREDRAW style is to be used in the first instance but not in the second. For more information, see WM_CALCVVALIDRECTS. |
| <b>CS_SYNCPAINT</b>    | Window is synchronously repainted. This style causes WS_SYNCPAINT to be set for all windows of this class.  |
| <b>CS_MOVENOTIFY</b>   | This class style should be used by a child window if it wants to be notified with a WM_MOVE message when its parent is moved. For more detail, see the WM_MOVE message description.   |
| <b>CS_CLIPCHILDREN</b> | Causes a window of style WS_CLIPCHILDREN to be created, regardless of whether this style bit is specified on the create window function.  |
| <b>CS_CLIPSIBLINGS</b> | Causes a window of style WS_CLIPSIBLINGS to be created, regardless of whether this style bit is specified on the create window function.  |
| <b>CS_PARENTCLIP</b>   | Causes a window of style WS_PARENTCLIP to be created, regardless of whether this style bit is specified on the create window function.  |
| <b>CS_SAVEBITS</b>     | Causes a window of style WS_SAVEBITS to be created, regardless of whether this style bit is specified on the create window function.  |
| <b>CS_PUBLIC</b>       | Causes a public window class to be registered. It is an error if this parameter is specified on any process other than the shell process.   |
| <b>CS_HITTEST</b>      | <p>If set, causes a WM_HITTEST message to be sent to the window, before sending any pointing device message.</p> <p>If not set, no WM_HITTEST message is sent, and it is assumed that the window returns HT_NORMAL if the window is not disabled, and HT_ERROR if the window is disabled.</p> <p>Top-level frame windows do not have CS_HITTEST set.</p>  |
| <b>CS_FRAME</b>        | If set, all windows of this class are expected to behave as frame windows.  |

## Window Styles

These window styles are available:

### **WS\_SYNCPAINT**

Window is synchronously repainted.

This style is set for windows that have Class Style CS\_SYNCPAINT. Applications can then turn this style on and off to vary the window processing.

System-Provided Window Styles:

### **WS\_ANIMATE**

This specifies that window animation will be turned on. Windows animation is a visual effect that occurs when the window is opened or closed; the window seems to zoom out when it is opened, and zoom in when it is closed.

This visual effect also depends on the Animation setting in the System-Settings notebook. If Animation is enabled and this window style is set, window animation occurs when the window is opened or closed. When Animation is disabled in the System-Settings notebook, this style has no effect and no window animation occurs.

### **WS\_CLIPCHILDREN**

This specifies that the area occupied by the children of a window is to be excluded when drawing in that window. Normally, it is included.

### **WS\_CLIPSIBLINGS**

This specifies that the area occupied by the siblings of a window is to be excluded when drawing in that window. Normally, it is included.

### **WS\_DISABLED**

This specifies that the window is disabled. The default is enabled.

### **WS\_MAXIMIZED**

This specifies that the frame window is to be created maximized.

When a window is moved or sized in the normal way at least one border should remain on the screen. When a window is maximized and the maximum size is as large as the screen all borders should be positioned just outside the screen.

### **WS\_MINIMIZED**

This specifies that the frame window is to be created minimized.

### **WS\_PARENTCLIP**

This controls how a window is clipped when a drawing action takes place into the window.

Generally, a WS\_PARENTCLIP window is not to draw outside its window rectangle.

### **WS\_SAVEBITS**

This specifies that the screen image of the area under a window of this style be saved when the window is made visible.

**WS\_VISIBLE**

This specifies that the window is visible. The default is invisible.

**Note:** A window can still be visible, in this sense, even if it cannot be seen because it is covered by other windows.

## Styles for Windows in Dialogs

**WS\_GROUP**

This identifies the dialog items that make up a group.

This style is to be specified on the first window of any group. Subsequent windows of the group must not have this style. The windows of the group must be adjacent siblings. This can be done by listing the windows consecutively in templates (see "Dialog Template" on page 31-24) or by inserting each new window in the group behind the previous one (WinCreateWindow).

**WS\_TABSTOP**

This identifies a dialog item as one to which the operator can TAB.

---

## General Window Messages

This section describes the window procedure actions upon receiving the following messages.

---

### PL\_ALTERED

This message is broadcast to all frame windows when the PrfReset function is issued.

#### Parameters

**param1**

**hiniUser** (HINI)

Handle of the new user profile.

**param2**

**hiniSystem** (HINI)

Handle of the new system profile.

#### Returns

**ulReserved** (ULONG)

Reserved value, must be 0.

#### Remarks

Applications should refresh their defaults from the user or system profile.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

### WM\_ACTIVATE

This message occurs when an application causes the activation or deactivation of a window.

#### Parameters

**param1**

**usactive** (USHORT)

Active indicator.

TRUE    The window is being activated

FALSE   The window is being deactivated.

## param2

### hwnd (HWND)

Window handle.

In the case of activation, *hwnd* identifies the window being activated. In the case of deactivation, *hwnd* identifies the window being deactivated.

## Returns

### ulReserved (ULONG)

Reserved value, should be 0.

## Remarks

A deactivation message (that is, a WM\_ACTIVATE message with *usactive* set to FALSE) is sent first to the window procedure of the main window being deactivated, before an activation message (that is, a WM\_ACTIVATE message with *usactive* set to TRUE) is sent to the window procedure of the main window being activated.

Any WM\_SETFOCUS messages with *usfocus* set to FALSE, are sent before the deactivation message. Any WM\_SETFOCUS messages with *usfocus* set to TRUE, are sent after the activation message.

If WinSetFocus is called during the processing of a WM\_ACTIVATE message, a WM\_SETFOCUS message with *usfocus* set to FALSE is not sent, as no window has the focus.

If a window is activated before any of its children have the focus, this message is sent to the frame window or to its FID\_CLIENT, if it exists.

**Note:** Except in the instance of a WM\_ACTIVATE message, with *usactive* set to TRUE, an application processing a WM\_ACTIVATE, or a WM\_SETFOCUS message should not change the focus window or the active window. If it does, the focus and active windows must be restored before the window procedure returns from processing the message. For this reason, any dialog boxes or windows brought up during the processing of a WM\_ACTIVATE, or a WM\_SETFOCUS message should be system modal.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_ACTIVATE (in Frame Controls)
- WM\_ACTIVATE (Language Support Dialog)
- WM\_ACTIVATE (Language Support Window)

---

## WM\_APPTERMINATENOTIFY

This message is posted when an application (started by another application) terminates.

### Parameters

#### param1

**happ** (HAPP)

Application handle.

#### param2

**flretcode** (ULONG)

Return code from the terminating application.

### Returns

**ulReserved** (ULONG)

Reserved value, must be 0.

### Remarks

The WM\_APPTERMINATENOTIFY message provides the capability for the starting application to be notified when the started application terminates.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_ADJUSTWINDOWPOS

This message is sent by the WinSetWindowPos call to enable the window to adjust its new position or size whenever it is about to be moved.

### Parameters

#### param1

**pswp** (PSWP)

SWP structure pointer.

The structure has been filled in by the WinSetWindowPos function with the proposed move or size data. The control can adjust this new position by changing the contents of the SWP structure. It can change the *x* or *y* fields to adjust its new position; or the *cx* or *cy* fields to adjust its new size, or the *hwndInsertBehind* field to adjust its new z-order.

## param2

**fizero** (ULONG)  
Zero.

## Returns

**flResult** (ULONG)

Window-adjustment status indicators.

These indicators are passed on to the WM\_WINDOWPOSCHANGED message that is sent after the window state change has occurred. Bits 0 through 15 of this parameter are reserved for system use and bits 16 through 31 are available for application use.

|                |  |
|----------------|--|
| 0              | No changes have been made              |
| AWP_MINIMIZED  | The frame window has been minimized.   |
| AWP_MAXIMIZED  | The frame window has been maximized.   |
| AWP_RESTORED   | The frame window has been restored.    |
| AWP_ACTIVATE   | The frame window has been activated.   |
| AWP_DEACTIVATE | The frame window has been deactivated. |

## Remarks

Frame controls can respond to this message to reposition themselves or resize themselves in the window frame.

Menu controls respond to this message as follows:

**MS\_ACTIONBAR not specified:** The SWP cx and SWP cy fields are set so that the menu window exactly contains all of the items in the menu. The SWP x and SWP y fields are not changed.

**MS\_ACTIONBAR specified and MS\_TITLEBUTTON not specified:** The items in the menu are arranged such that all of the items are visible within the width specified by the SWP cx field. This formatting may cause the menu items to be arranged in multiple lines. The SWP cx field is set to include all of the lines of the menu. The SWP x and SWP y fields are not changed.

**MS\_ACTIONBAR specified and MS\_TITLEBUTTON specified:** The SWP cx value is set to the accumulated width of the items in the menu. The height specified in the SWP cy field is not changed. In both instances, the SWP cx and SWP cy fields are only altered if SWP\_SIZE is specified in the fl field. Instead, the width of MS\_TITLEBUTTON menus is determined by the accumulated width of the items in the menu.

A list box does two things:

- Changes the height so as to accommodate an exact number of items.
- Automatically outsets its border. This means, for example, that the x, y, width, and height fields in the resource file specify the working area of the listbox. The border is drawn outside this area.

The entry field control, if `ES_MARGIN` is specified, outsets its margin. This means that in the resource file, the numbers specified as the `x-`, and `y-` position of an entry field control are taken to be the position where the first character of text is drawn, not where the lower-left corner of the surrounding box is drawn. Similarly, the height and width parameters apply to the editable area of the control; consequently, they do not include the margin.

When a dialog is created with `WinCreateDlg` or `WinLoadDlg`, a `WM_ADJUSTWINDOWPOS` message is sent to each child window after the dialog window is created, with a pointer to a `SWP` structure containing `fl` equal to `SWP_SIZE | SWP_MOVE` and the `x`, `y`, `cy`, and `cx` fields initialized to the current size and position of the window. The message enables the control to adjust its size or position, usually to compensate for its border, or margin, or both.

### Default Processing

The default window procedure takes no action on this message, other than to set `flResult` to 0.

---

## WM\_BEGINDRAG

This message occurs when the operator initiates a drag operation.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if `fPointer` is not set to `TRUE`.

**param2**

**fPointer** (USHORT)

Input device flag.

`TRUE`    Message resulted from pointer event  
`FALSE`   Message resulted from keyboard event.

**rc** (BOOL)

Processed indicator.

`TRUE`    Message processed  
`FALSE`   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value `SV_BEGINDRAG`.



## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to FALSE.

---

## WM\_BEGINSELECT

This message occurs when the operator initiates a swipe selection.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *fPointer* is not set to TRUE.

**param2**

**fPointer** (USHORT)

Input device flag.

TRUE Message resulted from pointer event

FALSE Message resulted from keyboard event.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE Message processed

FALSE Message ignored.

### Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value SV\_BEGINSELECT.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to FALSE.

---

## WM\_BUTTON1CLICK

This message occurs when the operator presses and then releases button 1 of the pointing device within a specified period of time, and without moving the mouse.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### **rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON1DBLCLK

This message occurs when the operator presses button 1 of the pointing device twice within a specified time, as detailed below.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

A double-click is recognized if all of the following are true:

- Two clicks are of the same button.
- No intervening pointing device button is pressed.
- The two clicks occur within the double-click time interval as defined by the SV\_DBLCLKTIME system value.

- The two clicks occur within a small spatial distance. This is defined by the rectangle, the length of whose sides parallel to the x- and y-axes are respectively, the SV\_CXDBLCLICK and SV\_CYDBLCLICK system values. The first click is assumed to be at the center of this rectangle.

The keyboard control codes specified by “flags” reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_BUTTON1DBLCLK (in Frame Controls)
- WM\_BUTTON1DBLCLK (in Multiline Entry Fields)

---

## WM\_BUTTON1DOWN

This message occurs when the operator presses pointer button one.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit test process, which determined the window to be associated with this message. For details of the possible values, see “WM\_HITTEST” on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

**KC\_NONE**            Indicates that no key is pressed.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

## Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

It is the responsibility of the application to ensure that the appropriate frame window is activated and that the focus is to the appropriate window, by using the `WinSetFocus` function. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

## Default Processing

The default window procedure activates the window using `WinSetActiveWindow`, and then sets `rc` to `FALSE`.

## Related Messages

- `WM_BUTTON1DOWN` (in Frame Controls)
- `WM_BUTTON1DOWN` (in Multiline Entry Fields)

---

## WM\_BUTTON1MOTIONEND

This message occurs when the operator completes a drag operation which was initiated by pressing button one on the pointing device.

## Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is terminated.

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see `WM_HITTEST`.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

## Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON1MOTIONSTART

This message occurs when the operator initiates a drag operation by moving the mouse while pressing button one on the pointing device.

## Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is started.

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

## Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON1UP

This message occurs when the operator releases button 1 of the pointing device.

### Parameters

**param1**

**ptspointerpos (POINTS)**

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

**param2**

**fsHitTestres (USHORT)**

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

**fsflags (USHORT)**

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

**rc (BOOL)**

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointing device button information. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message other than to set *rc* to FALSE.

## Related Messages

- WM\_BUTTON1UP (in Frame Controls)
- WM\_BUTTON1UP (in Multiline Entry Fields)

---

## WM\_BUTTON2CLICK

This message occurs when the operator presses and then releases button 2 of the pointing device within a specified period of time, and without moving the mouse.

## Parameters

### param1

#### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

### param2

#### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

#### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

## Returns

### **rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

## Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.



## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON2DBLCLK

This message occurs when the operator presses button 2 of the pointing device twice within a specified time, as detailed in “WM\_BUTTON1DBLCLK” on page 10-12.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see “WM\_HITTEST” on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### **rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information. The keyboard control codes specified by “flags” reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

## Default Processing

The default window procedure processes this message identically to WM\_BUTTON1DBLCLK.

## Related Messages

- WM\_BUTTON2DBLCLK (in Frame Controls)

---

## WM\_BUTTON2DOWN

This message occurs when the operator presses button 2 on the pointing device.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit test process, which determined the window to be associated with this message. For details of the possible values, see “WM\_HITTEST” on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

**KC\_NONE**            Indicates that no key is pressed.

### Returns

#### **rc** (BOOL)

Processed indicator.

**TRUE**    Message processed

**FALSE**   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointing device button information.

It is the responsibility of the application to ensure that the appropriate frame window is activated and that the focus is to the appropriate window, by using the `WinSetFocus` function. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure processes this message identically to "WM\_BUTTON1DOWN" on page 10-13.

### Related Messages

- WM\_BUTTON2DOWN (in Frame Controls)

---

## WM\_BUTTON2MOTIONEND

This message occurs when the operator completes a drag operation which was initiated by pressing button two on the pointing device.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is terminated.

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON2MOTIONSTART

This message occurs when the operator initiates a drag operation by moving the mouse while pressing button two on the pointing device.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is started.

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON2UP

This message occurs when the operator releases button 2 of the pointing device.

### Parameters

#### param1

##### ptspointerpos (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### fsHitTestres (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

##### fsflags (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE    Message processed  
FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointing device button information. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message other than to set *rc* to FALSE.

## Related Messages

- WM\_BUTTON2UP (in Frame Controls)

---

## WM\_BUTTON3CLICK

This message occurs when the operator presses and then releases button 3 of the pointing device within a specified period of time, and without moving the mouse.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

**fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.  
.\*>>> Removed per M.Ng      S.Kipp 7/22/94

### Returns

**rc** (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON3DBLCLK

This message occurs when the operator presses button 3 of the pointing device twice within a specified time, as detailed in "WM\_BUTTON1DBLCLK" on page 10-12.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer button information. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure processes this message identically to WM\_BUTTON1DBLCLK.

---

## WM\_BUTTON3DOWN

This message occurs when the operator presses button 3 on the pointing device.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit test process, which determined the window to be associated with this message. For details of the possible values, see “WM\_HITTEST” on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### **rc** (BOOL)

Processed indicator.

TRUE      Message processed

FALSE     Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointing device button information.

It is the responsibility of the application to ensure that the appropriate frame window is activated and that the focus is to the appropriate window, by using the WinSetFocus function. The keyboard control codes specified by “flags” reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure processes this message identically to “WM\_BUTTON1DOWN” on page 10-13.



---

## WM\_BUTTON3MOTIONEND

This message occurs when the operator completes a drag operation which was initiated by pressing button three on the pointing device.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is terminated.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON3MOTIONSTART

This message occurs when the operator initiates a drag operation by moving the mouse while pressing button three on the pointing device.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the hit-tested window, when the drag operation is started.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

### Returns

#### **rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_BUTTON3UP

This message occurs when the operator releases button 3 of the pointing device.

### Parameters

#### param1

##### **ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

#### param2

##### **fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see "WM\_HITTEST" on page 10-50.

##### **fsflags** (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

KC\_NONE            Indicates that no key is pressed.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE    Message processed

FALSE   Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointing device button information. The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

### Default Processing

The default window procedure processes this message identically to WM\_BUTTON1UP.

---

## WM\_CALCFRAMERECT

This message occurs when an application uses the WinCalcFrameRect function.

### Parameters

#### param1

##### pRect (PRECTL)

Rectangle structure.

This points to a RECTL structure.

#### param2

##### usFrame (USHORT)

Frame indicator.

TRUE     Frame rectangle provided  
FALSE    Client area rectangle provided.

### Returns

#### rc (BOOL)

Rectangle-calculated indicator.

TRUE     Successful completion  
FALSE    Error occurred or the calculated rectangle is empty.

### Remarks

This message is sent to the frame control to perform the appropriate calculation. If the low word of MP2 is TRUE, the RECTL structure in MP1 contains a frame window and this message calculates the RECTL of the client. If the low word of MP2 is FALSE, MP1 contains a client window and this message calculates the RECTL of the frame.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_CALCFRAMERECT (in Frame Controls)

---

## WM\_CALCVALIDRECTS

This message is sent from WinSetWindowPos and WinSetMultWindowPos to determine which areas of a window can be preserved if a window is sized, and which should be redisplayed.

### Parameters

**param1**

**pOldNew** (PRECTL)

Window-rectangle structures.

This points to two RECTL structures. The first structure contains the rectangle of the window before the move, the second contains the rectangle of the window after the move. The coordinates of the rectangles are relative to the parent window.

**param2**

**pNew** (PSWP)

New window position.

This points to a SWP structure that contains information about the window after it is resized (see the WinSetWindowPos function).

### Returns

**usAlign** (USHORT)

Alignment control.

This instructs WinSetWindowPos how to align valid window bits. This value is made up from CVR\_\* flags, as follows:

|                 |   |
|-----------------|---|
| CVR_ALIGNLEFT   | Align with the left edge of the window.   |
| CVR_ALIGNBOTTOM | Align with the bottom edge of the window.   |
| CVR_ALIGNTOP    | Align with the top edge of the window.  |
| CVR_ALIGNRIGHT  | Align with the right edge of the window.  |
| CVR_REDRAW      | The whole window is invalid. If CVR_REDRAW, is set, the whole window is assumed invalid, otherwise, the remaining flags can be ORed together to get different kinds of alignment. For example:<br>(CVR_ALIGNLEFT   CVR_ALIGNTOP)<br>aligns the valid window area with the top-left of the window. |
| 0               | It is assumed the application has changed the rectangles pointed to by <i>pOldNew</i> and <i>pNew</i> itself.   |

## Remarks

This message is *not* sent if this window has the `CS_SIZEREDRAW` style, indicating size-sensitive window content that must be totally redrawn if sized.

This enables the application to determine if the position of the window has changed as well as its size; this can aid alignment processing.

These rectangles can be modified by the window procedure to cause parts of the window to be redrawn and not preserved.

The window manager tries to preserve the screen image by copying the image described by the old rectangle into the image described by the new rectangle. In this way, an application can control the alignment of the preserved image as well, by changing the origin of the first rectangle.

If no change is made to either rectangle, the entire window area is preserved. If either rectangle is empty, the entire window area is completely redrawn by the operation.

**Note:** This functionality can be used to optimize window updating when the window is resized. For example, if the application returns that the window is to be aligned with the top-left corner, and the top border is sized, the screen data of the window moves with the top border.

In all instances, the rectangles are intersected with the area of the screen that is actually visible and the valid area of the window. That is, only the window area that contains window information is copied.

For example, consider an application that has two scroll bars, that are children of the client window. When the window is resized, the scroll bars must be completely redrawn. By returning rectangles that exclude the scroll bars, the area of the scroll bars is completely redrawn, thereby preserving only the part of the screen that is worth preserving.

## Default Processing

The default window procedure processing is to align the valid area with the top-left of the window by returning:

```
(CVR_ALIGNTOP | CVR_ALIGNLEFT)
```

In addition, any child windows intersecting the source rectangle pointed to by *pOldNew* of this message, are also offset with the aligned window area.

---

## WM\_CHAR

This message is sent when an operator presses a key.

### Parameters

param1

#### fsflags (USHORT)

Keyboard control codes.

|                |  |
|----------------|--|
| KC_CHAR        | Indicates that <i>usch</i> value is valid.   |
| KC_SCANCODE    | Indicates that <i>ucscancode</i> is valid.<br><br>Generally, this is set in all WM_CHAR messages generated from actual operator input. However, if the message has been generated by an application that has issued the WinSetHook function to filter keystrokes, or posted to the application queue, this may not be set. |
| KC_VIRTUALKEY  | Indicates that <i>usvk</i> is valid.<br><br>Normally <i>usvk</i> should be given precedence when processing the message.<br><br><b>Note:</b> For those using hooks, when this bit is set, KC_SCANCODE should usually be set as well.   |
| KC_KEYUP       | The event is a key-up transition; otherwise it is a down transition.   |
| KC_PREVDOWN    | The key has been previously down; otherwise it has been previously up.   |
| KC_DEADKEY     | The character code is a dead key. The application is responsible for displaying the glyph for the dead key without advancing the cursor.   |
| KC_COMPOSITE   | The character code is formed by combining the current key with the previous dead key.  |
| KC_INVALIDCOMP | The character code is not a valid combination with the preceding dead key. The application is responsible for advancing the cursor past the dead-key glyph and then, if the current character is not a space, sounding the alarm and displaying the new character code.  |
| KC_LONEKEY     | Indicates if the key is pressed and released without any other keys being pressed or released between the time the key goes down and up.   |
| KC_SHIFT       | The SHIFT state is active when key press or release occurred.  |
| KC_ALT         | The ALT state is active when key press or release occurred.  |

**KC\_CTRL**            The CTRL state was active when key press or release occurred.

**ucrepeat** (UCHAR)  
Repeat count.

**ucscancode** (UCHAR)  
Hardware scan code.

A keyboard-generated value that identifies the keyboard event. This is the raw scan code, not the translated scan code.

## **param2**

**usch** (USHORT)  
Character code.

The character value translation of the keyboard event resulting from the current code page that would apply if the CTRL or ALT keys were not depressed.

**usvk** (USHORT)  
Virtual key codes.

A virtual key value translation of the keyboard event resulting from the virtual key code table. The low-order byte contains the **vk** value, and the high-order byte is always set to zero by the standard translate table.

0    This value applies if *fsflags* does not contain **KC\_VIRTUALKEY**.

## **Returns**

**rc** (BOOL)  
Processed indicator.

TRUE    Message processed  
FALSE   Message ignored.

## **Remarks**

This message is posted to the queue associated with the window that has the focus.

The set of keys that causes a **WM\_CHAR** message is device-dependent.

When this message is processed, precedence should normally be given to a valid virtual key if there is one contained in the message.

There are several instances when a window procedure may receive this message with the **KC\_KEYUP** bit set, although it did not receive this message for the down transition of the key.

For example,

- The down transition of the key is translated by the function `WinTranslateAccel`, into a **WM\_COMMAND**, **WM\_SYSCOMMAND**, **WM\_HELP**, or a **WM\_NULL** message.



- The key down causes the input focus to change (tab to another window, dismiss a dialog, exit a program, and so on).
- Some other event happens that changes the focus between the time that the key is pressed down and the time that it is released.

Applications should normally only process WM\_CHAR messages that do not have the KC\_KEYUP bit set.

Except for the special instance where the LONEKEY flag is set on an accelerator key definition, all translations are done on the down stroke of the character.

When the current character is a double-byte character then *param2* contains both bytes of the double-byte character. These bytes are in the order CHAR1FROMMP, CHAR2FROMMP. When the current character is a single-byte character, CHAR2FROMMP contains 0.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message other than to set *rc* to FALSE.

## Related Messages

- WM\_CHAR (Default Dialogs)
- WM\_CHAR (in Entry Fields)
- WM\_CHAR (in Frame Controls)
- WM\_CHAR (in List Boxes)
- WM\_CHAR (in Multiline Entry Fields)

## Examples

This example uses the CHARMSG macro to process a WM\_CHAR message. It first uses the macro to determine if a key was released. It then uses the macro to generate a switch statement based on the character received.

```
MRESULT CALLBACK GenericWndProc(hwnd, usMessage, mp1, mp2)

HWND  hwnd;
USHORT usMessage;
MPARAM mp1;
MPARAM mp2;
{

    switch (usMessage) {
    case WM_CHAR:
        if (CHARMSG(&usMessage)->fs & KC_KEYUP) {
            switch (CHARMSG(&usMessage)->chr) {
```

---

## WM\_CHORD

This message occurs when the operator presses both button one and button two on the pointing device.

### Parameters

**param2**

**fsHitTestres** (USHORT)

Hit-test result.

*fsHitTestres* provides the hit-test result. It contains the value returned from the hit-test process, which determines the window to be associated with this message. For details of the possible values, see WM\_HITTEST.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that is to receive the pointer-button information.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_CLOSE

This message is sent to a frame window to indicate that the window is being closed by the user.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is sent by the frame to itself as a result of receiving a WM\_SYSCOMMAND message with SC\_CLOSE code set. If this message is passed to WinDefDlgProc, this function calls WinDismissDlg and passes the DID\_CANCEL result code to it.

## Default Processing

The default window procedure posts a WM\_QUIT message to the appropriate queue and sets *ulReserved* to 0.

## Related Messages

- WM\_CLOSE (Default Dialogs)
- WM\_CLOSE (in Frame Controls)

## Examples

In this example, the *fChanges* variable is checked. If it is TRUE, the user is asked if he wants to exit without saving any changes. If the user responds by choosing the No button, zero is returned and the application does not exit. If the user responds by choosing the Yes button, a WM\_QUIT message is posted and the application terminates.

```
case WM_CLOSE:
    if (fChanges) {
        if (WinMessageBox(HWND_DESKTOP, hwndClient,
            "Do you want to exit without saving your changes?",
            "", 0, MB_NOICON | MB_YESNO) == MBID_NO)
            return (0L);
    }
    WinPostMsg(hwnd, WM_QUIT, 0L, 0L);
    return (0L);
```

---

## WM\_COMMAND

This message occurs when a control has a significant event to notify to its owner, or when a key stroke has been translated by an accelerator table.

### Parameters

#### param1

##### **uscmd** (USHORT)

Command value.

It is the responsibility of the application to be able to relate *uscmd* to an application function.

#### param2

##### **ussource** (USHORT)

Source type.

Identifies the type of control:

|                           |  |
|---------------------------|--|
| <b>CMDSRC_PUSHBUTTON</b>  | Posted by a push-button control. <i>uscmd</i> is the window identity of the push button.             |
| <b>CMDSRC_MENU</b>        | Posted by a menu control. <i>uscmd</i> is the identity of the menu item.                             |
| <b>CMDSRC_ACCELERATOR</b> | Posted as the result of an accelerator. <i>uscmd</i> is the accelerator command value.               |
| <b>CMDSRC_FONTDLG</b>     | Font dialog. <i>uscmd</i> is the identity of the font dialog.  |
| <b>CMDSRC_FILEDLG</b>     | File dialog. <i>uscmd</i> is the identity of the file dialog.  |
| <b>CMDSRC_OTHER</b>       | Other source. <i>uscmd</i> gives further control-specific information defined for each control type. |

##### **uspointer** (USHORT)

Pointer-device indicator.

**TRUE** The message is posted as a result of a pointer-device operation.

**FALSE** The message is posted as a result of a keyboard operation.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is posted to the queue of the owner of the control.

WM\_Command handles popup menu command identifiers for pickup, putdown and cancel drag operations. It determines which items to display based on the state of the lazy drag and droppability of the lazy drag set.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_COMMAND (Default Dialogs)
- WM\_COMMAND (in Button Controls)
- WM\_COMMAND (in Menu Controls)
- WM\_SYSCOMMAND (in Title Bar Controls)

---

## WM\_CONTEXTMENU

This message occurs when the operator requests a pop-up menu.

## Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *fPointer* is not set to TRUE.

**param2**

**usReserved** (USHORT)

Reserved value, 0.

**fPointer** (USHORT)

Input device flag.

TRUE     Message resulted from keyboard event.

FALSE    Message resulted from mouse pointer event.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

## Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value `SV_CONTEXTMENU`, or a keyboard event, specified by the system value `SV_CONTEXTMENUMB`.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to `FALSE`.

---

## WM\_CONTROL

This message occurs when a control has a significant event to notify to its owner.

## Parameters

### param1

#### **id** (USHORT)

Control-window identity.

This is either the *id* parameter of the `WinCreateWindow` function or the identity of an item in a dialog template.

#### **usnotifycode** (USHORT)

Notify code.

The meaning of the notify code depends on the type of the control. For details, refer to the section describing that control.

### param2

#### **ulcontrolspect** (ULONG)

Control-specific information.

The meaning of the control-specific information depends on the type of the control. For details, refer to the section describing that control.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is sent to the owner of the control, thereby offering it the opportunity to perform some activity before returning to the control.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_CONTROL (in Button Controls)
- WM\_CONTROL (in Entry Fields)
- WM\_CONTROL (Language Support Dialog)
- WM\_CONTROL (Language Support Window)
- WM\_CONTROL (in List Boxes)
- WM\_CONTROL (in Multiline Entry Fields)
- WM\_CONTROL (in Combination Boxes)
- WM\_CONTROL (in Spin Button Controls)

---

## WM\_CONTROLPOINTER

This message is sent to a owner window of a control when the pointing device pointer moves over the control window, allowing the owner to set the pointing device pointer.

### Parameters

**param1**

**usidCtl** (USHORT)  
Control identifier.

**param2**

**hptrNew** (HPOINTER)  
Handle of the pointing device pointer that the control is to use.

### Returns

**hptrRet** (HPOINTER)

Returned pointing device-pointer handle that is then used by the control.

### Remarks

The recommended approach for an application, that does not have specific reasons for controlling the pointer appearance, is to pass the message to the default window procedure.

### Default Processing

The default window procedure returns *hptrNew*.

---

## WM\_CREATE

This message occurs when an application requests the creation of a window.

### Parameters

#### param1

##### **ctldata** (PVOID)

Pointer to control data.

This points to a Control-Data data structure initialized with the data provided in the *pCtlData* parameter of the *WinCreateWindow* function. This pointer is also contained in the *pCREATE* parameter.

This parameter **MUST** be a pointer rather than a long.

The first 2 bytes in the data referenced by this pointer should be the total size of the data referenced by the pointer, (for example, see the *ENTRYFDATA* or the *FRAMECDATA* structure). PM requires this information to enable it to ensure that the referenced data is accessible to both 16-bit and 32-bit code.

#### param2

##### **pCREATE** (PCREATESTRUCT)

Create structure.

This points to a *CREATESTRUCT* data structure. See the description of *ctldata* for a complete description.

### Returns

#### **rc** (BOOL)

Error indicator.

TRUE     Discontinue window creation  
FALSE    Continue window creation.

### Remarks

This message is sent to the window procedure of the window being created, thus offering it an opportunity to initialize that window.

The window procedure receives this after the window is created but before the window becomes visible.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to *FALSE*, which is equivalent to continuing the creation of the window.



---

## WM\_DESTROY

This message occurs when an application requests the destruction of a window.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is sent to the window procedure of the window being destroyed after it has been hidden on the device, thereby offering it an opportunity to perform some termination action for that window.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_DRAWITEM

This notification is sent to the owner of a control each time an item is to be drawn.

### Parameters

**param1**

**idIdentity** (USHORT)

Window identifier.

The window identity of the control sending this notification message.

## param2

### **ulcontrolspect** (ULONG)

Control-specific information.

The meaning of the control-specific information depends on the type of control. For details of each control type, refer to the appropriate section.

## **Returns**

### **rc** (BOOL)

Item-drawn indicator.

- TRUE     The owner has drawn the item, and so the control does not draw it.  
FALSE    If the item contains text and the owner does not draw the item, the owner returns this value and the control draws the item.

## **Remarks**

A control can only display some types of information, and emphasize items in a control-specific manner. Therefore, if special items are to be displayed or emphasized in a special manner, this must be done by the owner window of the control.

The control window procedure generates this message and sends it to the owner of the control, informing the owner that an item is to be drawn, offering the owner the opportunity to draw that item and to indicate that either the item has been drawn or that the control is to draw it.

## **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

## **Related Messages**

- WM\_DRAWITEM (in Frame Controls)
- WM\_DRAWITEM (in List Boxes)
- WM\_DRAWITEM (in Menu Controls)

---

## **WM\_ENABLE**

This message notifies a windows of a change to its enable state.

## **Parameters**

### **param1**

### **usnewenabledstate** (USHORT)

New enabled state indicator.

- TRUE     The window was set to the enabled state.  
FALSE    The window was set to the disabled state,

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is sent to the window procedure of the window whose enable state has been changed, thereby giving it an opportunity to perform some action appropriate to new state of the window.

This is just a notification message. If you want to change the enable state of a window, you would use `WinEnableWindow`

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- `WM_ENABLE` (in Button Controls)
- `WM_ENABLE` (in Multiline Entry Fields)

---

## WM\_ENDDRAG

This message occurs when the operator completes a drag operation.

## Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *fPointer* is not set to TRUE.

**param2**

**fPointer** (USHORT)

Input device flag.

TRUE Message resulted from pointer event

FALSE Message resulted from keyboard event.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE Message processed

FALSE Message ignored.

### Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value `SV_ENDDRAG`.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to `FALSE`.

---

## WM\_ENDSELECT

This message occurs when the operator either makes a selection or completes a swipe selection.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *fPointer* is not set to `TRUE`.

**param2**

**fPointer** (USHORT)

Input device flag.

TRUE Message resulted from pointer event

FALSE Message resulted from keyboard event.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE Message processed

FALSE Message ignored.

## Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value `SV_ENDSELECT`.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to `FALSE`.

---

## WM\_ERROR

This message occurs when an error is detected in a `WinGetMsg` or a `WinPeekMsg` function.

## Parameters

**param1**

**usererrorcode** (USHORT)

Error code.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The application can detect the error situation after the `WinGetMsg` or the `WinPeekMsg` function and before the `WinDispatchMsg` function.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_FOCUSCHANGE

This message occurs when the window possessing the focus is changed.

### Parameters

#### param1

##### **hwndFocus** (HWND)

Focus window handle.

#### param2

##### **usSetFocus** (USHORT)

Focus flag.

**TRUE** The window is receiving the focus and *hwndFocus* identifies the window losing the focus.

**FALSE** The window is losing the focus and *hwndFocus* identifies the window receiving the focus.

##### **fsFocusChange** (USHORT)

Focus changing indicators.

The indicators are passed from the *WinFocusChange* function.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is sent to both the windows gaining and losing the focus.

### Default Processing

The default window procedure sends this message to the owner or parent, if it exists and is not the desktop. Otherwise, it sets *ulReserved* to 0.

### Related Messages

- WM\_FOCUSCHANGE (in Frame Controls)

---

## WM\_FORMATFRAME

This message is sent to a frame window to calculate the sizes and positions of all of the frame controls and the client window.

### Parameters

#### param1

##### **pswp** (PSWP)

Structure array.

This points to an array that is to hold the SWP structures.

#### param2

##### **pprectl** (PRECTL)

Pointer to client window rectangle.

This is typically the window rectangle of *pswp*, but where the window has a wide border, as specified by *FCF\_DLGBORDER* for example, the rectangle is inset by the size of the border.

### Returns

#### **ccount** (USHORT)

Count of the number of SWP arrays returned.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ccount* to the default value of 0.

### Related Messages

- WM\_FORMATFRAME (in Frame Controls)

---

## WM\_HELP

This message occurs when a control has a significant event to notify to its owner or when a key stroke has been translated by an accelerator table into a WM\_HELP.

### Parameters

#### param1

##### **uscmd** (USHORT)

Command value.

It is the responsibility of the application to be able to relate *uscmd* to an application function.

#### param2

##### **ussource** (USHORT)

Source type.

Identifies the type of control:

|                    |  |
|--------------------|--|
| CMDSRC_PUSHBUTTON  | Posted by a push-button control. <i>uscmd</i> is the window identity of the push button.             |
| CMDSRC_MENU        | Posted by a menu control. <i>uscmd</i> is the identity of the menu item.                             |
| CMDSRC_ACCELERATOR | Posted as the result of an accelerator. <i>uscmd</i> is the accelerator command value.               |
| CMDSRC_OTHER       | Other source. <i>uscmd</i> gives further control-specific information defined for each control type. |

##### **uspointer** (USHORT)

Pointer-device indicator.

|       |  |
|-------|--|
| TRUE  | If the message is posted as a result of a pointer-device operation |
| FALSE | If the message is posted as a result of a keyboard operation.      |

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is identical to a WM\_COMMAND message, but implies that the application should respond to this message by displaying help information.

This message is posted to the queue of the owner of the control.



## Default Processing

The default window procedure sends this message to the parent window, if it exists and is not the desktop. Otherwise, it sets *ulReserved* to 0.

## Related Messages

- WM\_HELP (in Button Controls)
- WM\_HELP (in Menu Controls)

---

## WM\_HITTEST

This message is sent to determine which window is associated with an input from the pointing device.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulresult** (ULONG)

Hit-test indicator.

The application may return one of these values:

|                |  |
|----------------|--|
| HT_NORMAL      | The message should be processed as normal. A WM_MOUSEMOVE, WM_BUTTON2DOWN, or WM_BUTTON1DOWN message is posted to the window.                                |
| HT_TRANSPARENT | The part of the window underneath the pointer is transparent; hit-testing should continue on windows underneath this window, as if the window did not exist. |
| HT_DISCARD     | The message should be discarded; no message is posted to the application.  |
| HT_ERROR       | As HT_DISCARD, except that if the message is a button-down message, an alarm sounds and the window concerned is brought to the foreground.                   |

## Remarks

This message occurs when an application requests a message by issuing a `WinPeekMsg` or a `WinGetMsg` function.

If the message that is to be retrieved represents a pointer related event, this message is sent to a window to determine whether the message is in fact destined for that window.

This message is only sent if the window class has the `CS_HITTEST` style set.

**Note:** The handling of this message determines whether a disabled window can process pointing device events.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulresult* to `HT_ERROR` if the window is disabled, or to `HT_NORMAL` otherwise.

---

## WM\_HSCROLL

This message occurs when a horizontal scroll bar control has a significant event to notify to its owner.

### Parameters

#### param1

##### **usidentifier** (USHORT)

Scroll bar control window identifier.

#### param2

##### **slider** (SHORT)

Slider position.

0        Either the operator is not moving the slider with the pointer device, or for the instance where *uscmd* is `SB_SLIDERPOSITION` the pointer is outside the tracking rectangle when the button is released.

Other    Slider position.

##### **uscmd** (USHORT)

Command.

|                           |   |
|---------------------------|---|
| <code>SB_LINELEFT</code>  | Sent if the operator clicks on the left arrow of the scroll bar, or depresses the <code>VK_LEFT</code> key.       |
| <code>SB_LINERIGHT</code> | Sent if the operator clicks on the right arrow of the scroll bar, or depresses the <code>VK_RIGHT</code> key.     |
| <code>SB_PAGELEFT</code>  | Sent if the operator clicks on the area to the left of the slider, or depresses the <code>VK_PAGELEFT</code> key. |

|                   |   |
|-------------------|---|
| SB_PAGERIGHT      | Sent if the operator clicks on the area to the right of the slider, or depresses the VK_PAGERIGHT key.                      |
| SB_SLIDERPOSITION | Sent to indicate the final position of the slider.  |
| SB_SLIDERTRACK    | If the operator moves the scroll bar slider with the pointer device, this is sent every time the slider position changes.   |
| SB_ENDSCROLL      | Sent when the operator has finished scrolling, but only if the operator has not been doing any absolute slider positioning. |

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_HSCROLL (in Horizontal Scroll Bars)

## WM\_INITDLG

This message occurs when a dialog box is being created.

## Parameters

**param1**

**hwnd** (HWND)

Focus window handle.

The handle of the control window that is to receive the input focus.

**param2**

**pcreate** (PVOID)

Application-defined data area.

This points to the data area and is passed by the WinLoadDlg, WinCreateDlg, and WinDlgBox functions in their *pCreateParams* parameter.

This parameter MUST be a pointer rather than a long.

The first 2 bytes in the data referenced by this pointer should be the total size of the data referenced by the pointer, (for example, see the ENTRYFDATA or the FRAMECDATA structure). PM requires this information to enable it to ensure that the referenced data is accessible to both 16-bit and 32-bit code.

## Returns

**rc** (BOOL)

Focus set indicator.

**TRUE** Focus window is changed. The dialog procedure can change the window to receive the focus, by issuing a `WinSetFocus` whose *hwndNewFocus* specifies the handle of another control within the dialog box.

**FALSE** Focus window is not changed.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to **FALSE**.

## Related Messages

- `WM_INITDLG` (Default Dialogs)

---

## WM\_INITMENU

This message occurs when a menu control is about to become active.

## Parameters

**param1**

**smenuid** (SHORT)

Menu-control identifier.

**param2**

**hwnd** (HWND)

Menu-window handle.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- `WM_INITMENU` (in Frame Controls)
- `WM_INITMENU` (in Menu Controls)

---

## WM\_JOURNALNOTIFY

This message is used to maintain correct operation during journal playback.

### Parameters

#### param1

##### **ulCommand** (ULONG)

Command to journal.

JRN\_QUEUESTATUS    The WinQueryQueueStatus command must be journaled.

JRN\_PHYSKEYSTATE    The WinGetPhysKeyState command must be journaled.

#### param2

Data.

##### **fsQueueStatus** (USHORT)

Queue status.

See the *Summary* parameter of the WinQueryQueueStatus function.

##### **usScanCode** (USHORT)

Scan code.

See the *sc* parameter of the WinGetPhysKeyState function.

*param2* contains *usScanCode* and *usKeyState* if *ulCommand* has the value JRN\_PHYSKEYSTATE.

##### **usKeyState** (USHORT)

Key State.

See the *lKeyState* parameter of the WinGetPhysKeyState function.

*param2* contains *usScanCode* and *usKeyState* if *ulCommand* has the value JRN\_PHYSKEYSTATE.

### Returns

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

If the WinQueryQueueStatus or the WinGetPhysKeyState functions have new information since the last time they were called and there is a journal record hook installed, the journal record hook is called with this message to record this new information.

During playback, this message is interpreted by the system and the appropriate state restored.

Data values of the *param2* parameter depend on which command is to be journaled.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ulReserved* to 0.

---

## WM\_MATCHMNEMONIC

This message is sent by the dialog box to a control window to determine whether a typed character matches a mnemonic in its window text.

### Parameters

**param1**

**usmatch** (USHORT)  
Match character.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Returns

**rc** (BOOL)

Match indicator.

TRUE     Mnemonic found  
FALSE    Mnemonic not found, or an error occurred.

### Default Processing

The default dialog procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_MATCHMNEMONIC (in Button Controls)
- WM\_MATCHMNEMONIC (Default Dialogs)
- WM\_MATCHMNEMONIC (in Static Controls)

---

## WM\_MEASUREITEM

This notification is sent to the owner of a specific control to establish the height and width for an item in that control.

### Parameters

**param1**

**sidentity** (SHORT)  
Control identifier.

**param2**

**ulControlSpec** (ULONG)

Control-specific information.

The meaning of the control-specific information depends on the type of control. For details of each control type, refer to the appropriate control section.

## Returns

**ReturnCode**

**sHeight** (SHORT)

Height of item.

**sWidth** (SHORT)

Width of item.

## Remarks

When the owner receives this message, it must calculate and return the height and width (for a horizontally-scrollable list box control) of an item to the control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ReturnCode* to the default value of 0.

## Related Messages

- WM\_MEASUREITEM (in Frame Controls)
- WM\_MEASUREITEM (in List Boxes)
- WM\_MEASUREITEM (in Menu Controls)

---

## WM\_MENUEND

This message occurs when a menu control is about to terminate.

## Parameters

**param1**

**usmenuid** (USHORT)

Menu-control identifier.

**param2**

**hwnd** (HWND)

Menu-control window handle.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_MENUEND (in Menu Controls)

---

## WM\_MENUSELECT

This message occurs when a menu item has been selected.

## Parameters

**param1**

**usItem** (USHORT)

Identifier of selected item.

**usPostCommand** (USHORT)

Post-command flag.

**TRUE** Indicates that either a WM\_COMMAND, WM\_SYSCOMMAND, or WM\_HELP message is being posted by the menu control on return from the owner, subject to *rc*.

**FALSE** Indicates that no message is being posted by the menu control on return from the owner, subject to *rc*.

**param2**

**hwnd** (HWND)

Menu-control window handle.

## Returns

**rc** (BOOL)

Post indicator.

**TRUE** Indicates that either a WM\_COMMAND, WM\_SYSCOMMAND, or WM\_HELP message is to be posted by the menu control window procedure. The menu is dismissed if the selected item does not have a style of MIA\_NODISMISS.

**FALSE** Indicates that no message is to be posted by the menu control window procedure and that the menu is not dismissed.



## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to TRUE.

## Related Messages

- WM\_MENUSELECT (in Frame Controls)
- WM\_MENUSELECT (in Menu Controls)

---

## WM\_MINMAXFRAME

This message is sent to a frame window that is being minimized, maximized, or restored.

### Parameters

#### param1

##### *pswp* (PSWP)

Set window position structure.

This points to a SWP structure. The structure has the appropriate SWP\_\* indicators set to describe the operation that is occurring to the window.

#### param2

##### *ulReserved* (ULONG)

Reserved value, should be 0.

### Returns

#### *rc* (BOOL)

Processed indicator.

TRUE The message has been processed; the default system actions for the operation specified by the *pswp* parameter to the window are not to be performed.

FALSE The message has been ignored; the default system actions for the operation specified by the *pswp* parameter to the window are to be performed.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_MINMAXFRAME (in Frame Controls)

---

## WM\_MOUSEMAP

This message is specific to version 2.1, or higher, of the OS/2 operating system.

This message is used only by applications that wish to remap mouse messages in the PM input queue. It is not recommended for general application usage, and applications should NOT process this message in their window procedures.

### Parameters

**param1**

**ulPhysButton** (ULONG)

The physical button number (1, 2, or 3).

**param2**

**ulMappedButton** (ULONG)

The button to be mapped to (1, 2, or 3).

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

PM will interpret this message when it is read from the PM input queue, as a request to remap all subsequent mouse events for the desired button, until another WM\_MOUSEMAP message is received, cancelling that remap request. This message has no meaning to an application.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_MOUSEMOVE

This message occurs when the pointing device pointer moves.

### Parameters

**param1**

**sxMouse** (SHORT)

Pointing device x-coordinate.

**syMouse** (SHORT)

Pointing device y-coordinate.

## param2

### uswHitTest (USHORT)

Message result.

- Zero A pointing device capture is currently in progress
- Other The result of the WM\_HITTEST message.

### fsflags (USHORT)

Keyboard control codes.

In addition to the control codes described with the WM\_CHAR message, the following keyboard control codes are valid.

- KC\_NONE Indicates that no key is pressed

## Returns

### rc (BOOL)

Processed indicator.

- TRUE The window procedure did process the message.
- FALSE The window procedure did not process the message.

## Remarks

The keyboard control codes specified by "flags" reflects the keyboard state at the time the mouse message was initiated. This may or may not reflect the current keyboard state.

*param1* contains the position of the pointing device in window coordinates relative to the bottom-left corner of the window.

## Default Processing

The default window procedure sets the pointer shape using the WinSetPointer function and sets *rc* to FALSE.

## Related Messages

- WM\_MOUSEMOVE (in Multiline Entry Fields)

---

## WM\_MOVE

This message occurs when a window with style CS\_MOVENOTIFY changes its absolute position.

## Parameters

### param1

#### ulReserved (ULONG)

Reserved value, should be 0.

## **param2**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Remarks**

The message is sent from `WinSetWindowPos`, `WinSetMultWindowPos`, and `WinScrollWindow`.

The message is sent to any window when it is moved relative to its parent window. In addition, a `WM_MOVE` message is also sent to any children of that window that have style `CS_MOVENOTIFY`.

The new position of the window is obtained by calling `WinQueryWindowRect`, and can make those rectangle coordinates relative to any window by calling `WinMapWindowPoints`.

**Note:** There are several instances where windows have cause to know if they have been moved, and these include the occasions when the window does not change position relative to its parent, but does change position relative to the screen (its absolute position).

An example is menus. When a top-level menu control (child of the frame window) moves its absolute position as a result of the frame window being moved, the top-level menu control causes the movement of any pull-down menus along with its movement. The same applies to application/dialog box positional grouping. In some instances, a dialog box might cause to be moved as the main window is moved, to make room for other applications.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## **WM\_MSGBOXDISMISS**

This message notifies the owner of the message when a non-modal message box has been dismissed (the message box is no longer visible).

### **Parameters**

**param1**

**hwnd** (HWND)

Non-modal window handle.

**param2**

**ulButtonId** (ULONG)

Identity of the selected button in the message box.

### **Returns**

**ulReserved** (ULONG)

Reserved value, must be 0.

### **Remarks**

This message is processed within the owner's window procedure when a non-modal message box is dismissed. It is up to the parent to destroy the message box.

---

## **WM\_MSGBOXINIT**

This message notifies the owner of the message when a non-modal message box has been created and is currently being displayed.

### **Parameters**

**param1**

**hwnd** (HWND)

Non-modal window handle.

**param2**

**idWindow** (LONG)

Window identity of the message box.

### **Returns**

**ulReserved** (ULONG)

Reserved value, must be 0.

## Remarks

This message is processed within the owner's window procedure when a non-modal `WinMessageBox2` is created. It is up to the owner to store the window handle returned by this function. This handle is then used to properly destroy the message box when `WM_MSGBOXDISMISS` is received or when the parent chooses to destroy it.

---

## WM\_NEXTMENU

This message occurs when either the beginning or the end of the menu is reached by use of the cursor control keys.

### Parameters

**param1**

**hwndMenu** (HWND)

Menu-control window handle.

**param2**

**usPrev** (USHORT)

Previous-menu indicator.

TRUE Beginning of the menu has been reached

FALSE End of the menu has been reached.

### Returns

**hwndNewMenu** (HWND)

New menu window handle.

NULLHANDLE No new menu

Other New menu window handle.

### Default Processing

The default window procedure takes no action on this message, other than to set *hwndNewMenu* to `NULLHANDLE`.

### Related Messages

- `WM_NEXTMENU` (in Frame Controls)
- `WM_NEXTMENU` (in Menu Controls)

---

## **WM\_NULL**

This message is posted to activate message queues or modal loops.

### **Parameters**

#### **param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### **param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Remarks**

On receiving this message, the application should simply let the default processing take place.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_OPEN

This message occurs when the operator makes an *OPEN* request.

### Parameters

#### param1

##### usPointer (USHORT)

Input device flag.

TRUE Message resulted from pointer event

FALSE Message resulted from keyboard event

#### param2

##### ptspointerpos (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *usPointer* is not set to TRUE.

### Returns

#### rc (BOOL)

Processed indicator.

TRUE Message processed

FALSE Message ignored.

### Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value *SV\_OPEN*.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to FALSE.

---

## WM\_PACTIVATE

This message is posted when the Language Support Window or Dialog Procedure processes a *WM\_ACTIVATE* message.

### Parameters

#### param1

##### usactive (USHORT)

Active indicator.



TRUE The window was activated  
FALSE The window was deactivated.

## param2

### hwnd (HWND)

Window handle.

In the case of activation, *hwnd* identifies the window which was activated. In the case of deactivation, *hwnd* identifies the window which was deactivated.

## Returns

### ulReserved (ULONG)

Reserved value, should be 0.

## Remarks

The activation change has already occurred when the application receives this message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_PAINT

This message occurs when a window needs repainting.

## Parameters

### param1

#### ulReserved (ULONG)

Reserved value, should be 0.

### param2

#### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### ulReserved (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure issues the *WinBeginPaint* and *WinEndPaint* functions, and then sets *ulReserved* to 0.

## Related Messages

- WM\_PAINT (in Frame Controls)
- WM\_PAINT (Language Support Dialog)
- WM\_PAINT (Language Support Window)

## Examples

This example shows how an application gets a presentation space for drawing by calling the `WinBeginPaint` function. When drawing is complete, the `WinEndPaint` function is called to release the presentation space.

```
case WM_PAINT:
    hps = WinBeginPaint(hwnd, NULL, &rc1);
    .
    . /* drawing routines would go here */
    .
    WinEndPaint(hps);
    return (0L);
```

---

## WM\_PCONTROL

This message is posted when the Language Support Window or Dialog Procedure processes a `WM_CONTROL` message.

### Parameters

#### param1

##### **id** (USHORT)

Control-window identity.

This is either the *id* parameter of the `WinCreateWindow` function or the identity of an item in a dialog template.

##### **usnotifycode** (USHORT)

Notify code.

The meaning of the notify code depends on the type of the control. For details, refer to the section describing that control.

#### param2

##### **ulZero** (ULONG)

Zero.

- 0 The control-specific information in *ulcontrolspect* of the `WM_CONTROL` message is not available because the information might not be valid when the application receives this message.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The notification from the control has already been processed when the application receives this message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_PPAINT

This message is posted when the Language Support Window or Dialog Procedure processes a WM\_PAINT message.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure issues the WinBeginPaint and WinEndPaint functions, and then sets *ulReserved* to 0.

## Related Messages

- WM\_PPAINT (Language Support Dialog)
- WM\_PPAINT (Language Support Window)

---

## WM\_PRESPARAMCHANGED

This message is sent when a presentation parameter is set or removed dynamically from a window instance using the `WinSetPresParam` or `WinRemovePresParam` functions. It is also sent to all windows owned by the window whose presentation parameter was changed.

### Parameters

**param1**

**idAttrType** (ULONG)

Presentation parameter attribute identity.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message notifies a control when an inherited presentation parameter changes.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_PSETFOCUS

This message is posted when the Language Support Window or Dialog Procedure processes a `WM_SETFOCUS` message.

### Parameters

**param1**

**hwnd** (HWND)

Focus-window handle.

NULLHANDLE No window lost or received the focus.

Other Window handle.

## param2

### **usfocus** (USHORT)

Focus flag.

**TRUE** The window received the focus. *hwnd* is the window handle of the window which lost the focus, or `NULLHANDLE` if no window previously had the focus.

**FALSE** The window lost the focus. *hwnd* is the window handle of the window which received the focus, or `NULLHANDLE` if no window received the focus.

## **Returns**

### **ulReserved** (ULONG)

Reserved value, should be 0.

## **Remarks**

The focus change has already occurred when the application receives this message.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## **WM\_PSIZE**

This message is posted when the Language Support Window or Dialog Procedure processes a `WM_SIZE` message.

## **Parameters**

### **param1**

#### **scxold** (SHORT)

Old horizontal size.

#### **scyold** (SHORT)

Old vertical size.

### **param2**

#### **scxnew** (SHORT)

New horizontal size.

#### **scynew** (SHORT)

New vertical size.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The size change has already occurred when the application receives this message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SYSCOLORCHANGE

This message is posted when the Language Support Window or Dialog Procedure processes a WM\_SYSCOLORCHANGE message.

## Parameters

**param1**

**flOptions** (ULONG)

Options.

Copied from the *flOptions* parameter of the WinSetSysColors function.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

All windows in the system are invalidated so that they will be redrawn with the new system color.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_QUERYACCELTABLE

This message returns the handle to the accelerator table of a window.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**haccel** (HACCEL)

Accelerator table handle.

NULLHANDLE No accelerator table is associated with the window.

Other The handle of the accelerator table associated with the window.

### Default Processing

The default window procedure takes no action on this message, other than to set *haccel* to NULLHANDLE.

---

## WM\_QUERYCONVERTPOS

This message is sent by an application to determine whether it is appropriate to begin conversion of DBCS characters.

### Parameters

**param1**

**pCursorPos** (PRECTL)

Cursor position.

If *usCode* = QCP\_CONVERT, *pCursorPos* should be updated to contain the position of the cursor in the window receiving this message. The position is specified as a rectangle in screen coordinates.

If *usCode* = QCP\_NOCONVERT, *pCursorPos* should not be updated.

## param2

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **usCode** (USHORT)

Conversion code.

- |               |   |
|---------------|---|
| QCP_CONVERT   | Conversion may be performed for the window with the input focus, <i>pCursorPos</i> has been updated to contain the position of the cursor.  |
| QCP_NOCONVERT | Conversion should not be performed, the window with the input focus cannot receive DBCS characters, <i>pCursorPos</i> has not been updated. |

## Remarks

This message enables a DBCS application to determine whether the window with the input focus can handle DBCS characters. The *pCursorPos* parameter can be used as a guide for positioning any conversion window that the application requires.

## Default Processing

The default window procedure returns QCP\_CONVERT, and updates *pCursorPos* to the following values:

- xleft = -1
- ybottom = -1
- xright = 0
- ytop = 0

## Related Messages

- WM\_QUERYCONVERTPOS (in Button Controls)
- WM\_QUERYCONVERTPOS (in Title Bar Controls)
- WM\_QUERYCONVERTPOS (in Entry Fields)
- WM\_QUERYCONVERTPOS (in Frame Controls)
- WM\_QUERYCONVERTPOS (in List Boxes)
- WM\_QUERYCONVERTPOS (in Menu Controls)
- WM\_QUERYCONVERTPOS (in Scroll Bars)
- WM\_QUERYCONVERTPOS (in Static Controls)



---

## WM\_QUERYHELPINFO

This message returns the help instance associated with a frame window.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**lhelpinfo** (LONG)

Help information.

0 No help information associated with the window.

Other The help information associated with the window.

### Default Processing

The default window procedure takes no action on this message, other than to set *lhelpinfo* to 0.

---

## WM\_QUERYTRACKINFO

The frame control generates this message on receiving a WM\_TRACKFRAME (in Frame Controls) message.

### Parameters

**param1**

**ustflags** (USHORT)

Tracking flags.

Contains a combination of one or more TF\_\* flags as defined in the TRACKINFO structure.

**param2**

**ptrackinfo** (PTRACKINFO)

Track information structure.

This points to a TRACKINFO structure. The receiver of this message must modify this structure.

## Returns

**rc** (BOOL)

Continue indicator.

TRUE Continue sizing or moving

FALSE Terminate sizing or moving.

## Remarks

This message is sent to the window procedure of the owner of a frame control or title bar control respectively.

The TRACKINFO data structure specified by the *ptrackinfo* parameter is not initialized before the message is sent. It must be correctly completed before returning.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_TRACKFRAME (in Title Bar Controls)

---

## WM\_QUERYWINDOWPARAMS

This message occurs when an application queries the window parameters.

## Parameters

**param1**

**pwndparams** (PWNDPARAMS)

Window parameter structure.

This points to a window parameter structure; see “WNDPARAMS” on page A-207.

The valid values of *fsStatus* are WPM\_CCHTEXT, WPM\_TEXT, WPM\_CBCTLDATA, and WPM\_CTLDATA.

The flags in *fsStatus* are cleared as each item is processed. If the call is successful, *fsStatus* is 0. If any item has not been processed, the flag for that item is still set.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

If this message is sent to a window of another process, the information in, or identified by, *wndparams* must be in memory shared by both processes.

## Default Processing

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure identified by the *wndparams* to 0, and sets *rc* to FALSE.

## Related Messages

- WM\_QUERYWINDOWPARAMS (in Button Controls)
- WM\_QUERYWINDOWPARAMS (in Entry Fields)
- WM\_QUERYWINDOWPARAMS (in Frame Controls)
- WM\_QUERYWINDOWPARAMS (in List Boxes)
- WM\_QUERYWINDOWPARAMS (in Menu Controls)
- WM\_QUERYWINDOWPARAMS (in Multiline Entry Fields)
- WM\_QUERYWINDOWPARAMS (in Scroll Bars)
- WM\_QUERYWINDOWPARAMS (in Static Controls)
- WM\_QUERYWINDOWPARAMS (in Title Bars)

---

## WM\_QUIT

This message is posted to terminate the application.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

It causes WinGetMsg to return *rc* set to FALSE, rather than to TRUE, as for all other messages.

**Note:** Applications that call WinPeekMsg rather than WinGetMsg should test explicitly for WM\_QUIT.

This message should not be dispatched to the default window procedure. The intent of this message is to cause the WinGetMsg loop to terminate.

Typically this message is posted by the application when the application exit command is selected from the action bar.

This message is also sent to all applications when the system is closing down. To reply to this, the application should either cancel the request by issuing an WinCancelShutdown function or close itself down by issuing a WinDestroyMsgQueue function.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Examples

In this example, a WM\_CLOSE message is received. If the *fChanges* flag is set, the application calls a function to determine if the user wants to save the changes before exiting. This function (called *QuerySaveFile* in this example) asks the user if he wants to save the changes. If the user selects OK, the changes are saved. If the user selects cancel, the function returns this value and the application continues normal execution. Otherwise, it posts a WM\_QUIT message to terminate the application.

```
case WM_CLOSE:
    if (fChanges) {
        /* changes have not been saved */
        if (QuerySaveFile(hwnd) == MB_CANCEL) {
            return (0L); /* do not exit after all */
        }
    }
    WinPostMsg(hwnd, WM_QUIT, 0L, 0L);
    return (0L);
```

---

## WM\_REALIZEPALETTE

This message is sent to an application whenever changes have been made to the display hardware physical color table as a result of another application calling WinRealizePalette.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The application should call WinRealizePalette if it has a palette, or pass it on to the default window procedure if it does not.

If the return value from WinRealizePalette is greater than 0, the application should invalidate its window to cause a repaint using the newly-realized palette.

### Default Processing

The default window procedure calls WinRealizePalette with a NULL *hps* parameter. This causes the default palette to be realized. If the return value from WinRealizePalette is greater than 0, the default window procedure invalidates the window, causing it to be repainted with the newly-realized palette.

---

## WM\_SAVEAPPLICATION

This message is sent by the system to notify an application to save its current state.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

When an application receives this message, it is expected to save its current state by any convenient method, for example, in a profile or in an auxiliary file.

It is the responsibility of the application to use the saved information, as appropriate, when it is resumed.

Even if the application processes this message, it should also pass it to the default window procedure, by using the `WinDefWindowProc` call.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SEM1

This message is sent or posted by an application.

## Parameters

**param1**

**flAccumBits** (ULONG)

Semaphore value.

The semaphore values from all the `WM_SEM1` messages posted to a queue, are accumulated by a logical-OR operation.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

If the message is posted, it is merged with any existing WM\_SEM1 message on the queue by combining the two *flAccumBits* values using a logical-OR operation.

The WM\_SEM1 messages are queued higher than any other type of message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Examples

In this example, a thread notifies the client window that it is about to terminate. It sends the constant THREAD3 as the *flFlags* parameter so that when the client window receives the message, it can tell which thread terminated.

```
#define THREAD1 1    /* bit #1 */
#define THREAD2 2    /* bit #2 */
#define THREAD3 4    /* bit #3 */
VOID FAR Thread3() {
    .
    .
    .
    WinPostMsg(hwndClient, WM_SEM1, (MPARAM) THREAD3, 0);
    DosExit(EXIT_THREAD, 0);
}
```

---

## WM\_SEM2

This message is sent or posted by an application.

### Parameters

**param1**

**flAccumBits** (ULONG)

Semaphore value.

The semaphore values from all the WM\_SEM2 messages posted to a queue, are accumulated by a logical-OR operation.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

If the message is posted, it is merged with any existing WM\_SEM2 message on the queue by combining the two *flAccumBits* values using a logical-OR operation.

The WM\_SEM2 messages are queued above WM\_SEM3 and WM\_SEM4 messages, and above any WM\_PAINT or WM\_TIMER messages generated by the system, but lower than any other message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SEM3

This message is sent or posted by an application.

## Parameters

**param1**

**flAccumBits** (ULONG)

Semaphore value.

The semaphore values from all the WM\_SEM3 messages posted to a queue, are accumulated by a logical-OR operation.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

If the message is posted, it is merged with any existing WM\_SEM3 message on the queue by combining the two *flAccumBits* values using a logical-OR operation.

The WM\_SEM3 messages are queued above WM\_SEM4 messages, and any WM\_TIMER messages generated by the system, but lower than any other message.



## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SEM4

This message is sent or posted by an application.

### Parameters

#### param1

##### **flAccumBits** (ULONG)

Semaphore value.

The semaphore values from all the WM\_SEM4 messages posted to a queue, are accumulated by a logical-OR operation.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

If the message is posted, it is merged with any existing WM\_SEM4 message on the queue by combining the two *flAccumBits* values using a logical-OR operation.

The WM\_SEM4 messages are queued lower than any other type of message.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SETACCELTABLE

This message establishes the window accelerator table to be used for translation, when the window is active.

### Parameters

#### param1

**haccelNew** (HACCEL)

New accelerator table.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_SETFOCUS

This message occurs when a window is to receive or lose the input focus.

### Parameters

#### param1

**hwnd** (HWND)

Focus-window handle.

NULLHANDLE No window is losing or receiving the focus.

Other Window handle.

## param2

### **usfocus** (USHORT)

Focus flag.

- TRUE The window is receiving the focus. *hwnd* is the window handle of the window losing the focus, or NULLHANDLE if no window previously had the focus.
- FALSE The window is losing the focus. *hwnd* is the window handle of the window receiving the focus, or NULLHANDLE if no window is receiving the focus.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is sent to the window receiving or losing the focus, thereby giving it the opportunity to perform some appropriate processing.

**Note:** Except in the instance of WM\_ACTIVATE, with *usactive* set to TRUE, an application processing WM\_SETFOCUS or WM\_ACTIVATE messages should not change the focus window or active window. If it does, the focus and active window must be restored before the application returns from processing the message. For this reason, any dialog boxes or windows brought up during the processing of WM\_SETFOCUS or WM\_ACTIVATE messages should be system modal.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_SETFOCUS (Language Support Dialog)
- WM\_SETFOCUS (Language Support Window)

---

## WM\_SETHelpINFO

This message sets the help instance associated with this frame window when the window is active.

## Parameters

### param1

### **hhelpinfo** (LONG)

New help information.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

## Returns

**rc** (BOOL)  
Success indicator.

TRUE Successful completion  
FALSE Error occurred.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_SETSELECTION

This message occurs when a window is selected or deselected.

## Parameters

**param1**

**usselection** (USHORT)  
Selection flag.  
  
TRUE The window is selected.  
FALSE The window is deselected.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)  
Reserved value, should be 0.

## Remarks

The window procedure is expected to highlight or unhighlight the selected item of the window, as appropriate.

This message is sent to a window when it loses the focus to another window that it does not own. It allows an application to remove the selection when the focus is removed to another application, but to keep it if, for example, the same application displays a dialog box.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SETWINDOWPARAMS

This message occurs when an application sets or changes the window parameters.

### Parameters

#### param1

##### **pwndparams** (PWNDPARAMS)

Window parameter structure.

This points to a window parameter structure; see "WNDPARAMS" on page A-207.

The valid values of *fsStatus* are WPM\_TEXT and WPM\_CTLDATA.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE    Successful operation  
FALSE   Error occurred.

### Remarks

If this message is sent to a window of another process, the information in, or identified by, *pwndparams* must be in memory shared by both processes.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_SETWINDOWPARAMS (in Button Controls)
- WM\_SETWINDOWPARAMS (in Entry Fields)
- WM\_SETWINDOWPARAMS (in Frame Controls)
- WM\_SETWINDOWPARAMS (in List Boxes)
- WM\_SETWINDOWPARAMS (in Menu Controls)
- WM\_SETWINDOWPARAMS (in Multiline Entry Fields)
- WM\_SETWINDOWPARAMS (in Scroll Bars)
- WM\_SETWINDOWPARAMS (in Static Controls)

- WM\_SETWINDOWPARAMS (in Title Bar Controls)

---

## WM\_SHOW

This message occurs when the WS\_VISIBLE state of a window is being changed.

### Parameters

**param1**

**usshow** (USHORT)

Show indicator.

TRUE Show the window

FALSE Hide the window.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The message is sent after the visibility state has changed.

In this context, the terms “shown” or “hidden” refer to the state of the WS\_VISIBLE style bit. This message is *not* sent when a window is obscured by other windows above it.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SINGLESELECT

This message occurs when the operator selects a single object.

### Parameters

**param1**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *usPointer* is not set to TRUE.

## param2

### usPointer (USHORT)

Input device flag.

TRUE     Message resulted from pointer event  
FALSE    Message resulted from keyboard event.

## Returns

### rc (BOOL)

Processed indicator.

TRUE     Message processed  
FALSE    Message ignored.

## Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from a mouse event, specified by the system value SV\_SINGLESELECT.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_SIZE

This message occurs when a window changes its size.

## Parameters

### param1

#### scxold (SHORT)

Old horizontal size.

#### scyold (SHORT)

Old vertical size.

### param2

#### scxnew (SHORT)

New horizontal size.

#### scynew (SHORT)

New vertical size.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is not sent by WinCreateWindow when a window is created, and so any size-related processing must be done during the WM\_CREATE message processing in this instance.

This message is sent after the window has been actually sized, but before any repainting has been done. Any resizing or repositioning of child windows that might be necessary as a result of the size change is usually done during the processing of this message.

**Note:** It is generally unwise to output to the window during the processing of this message, because the area drawn might be redrawn, after the WM\_SIZE processing is complete, by the WinSetWindowPos function.

The processing of this message for a window which is displaying an advanced VIO presentation space must be carried out by the default advanced VIO window procedure.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_SIZE (in Frame Controls)
- WM\_SIZE (Language Support Dialog)
- WM\_SIZE (Language Support Window)

---

## WM\_SUBSTITUTESTRING

This message is sent from the WinSubstituteStrings call.

## Parameters

### **param1**

#### **iindex** (USHORT)

Substitution index.

A value corresponding to the decimal character in the substitution phrase.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.



## Returns

### **pString** (PSZ)

String to be substituted.

This points to a string (character) buffer.

0 No substitution string

Other Substitution string.

## Remarks

The WinSubstituteStrings call has encountered a substitution phrase in a string. The substitution phrase takes the form "%<digit>," where <digit> is a single decimal character; that is, 0 through 9.

## Default Processing

The default window procedure takes no action on this message, other than to set *pString* to 0.

---

## WM\_SYSCOLORCHANGE

This message is sent to all main windows when a change is made to the system colors by the WinSetSysColors function.

## Parameters

### **param1**

#### **flOptions** (ULONG)

Options.

Copied from the *flOptions* parameter of the WinSetSysColors function and therefore specifies which palette has been changed.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

All windows are invalidated, so that they are redrawn with the new colors. When this message is received, applications that depend on the system colors can query the new color values with the WinQuerySysColor call.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_SYSCOLORCHANGE (Language Support Dialog)
- WM\_SYSCOLORCHANGE (Language Support Window)

---

## WM\_SYSCOMMAND

This message occurs when a control has a significant event to report to its owner or when a key stroke has been translated by an accelerator table.

### Parameters

#### param1

##### **uscmd** (USHORT)

Command value.

The command value can be one of the SC\_\* values. It is the responsibility of the application to be able to relate *uscmd* to an application function.

#### param2

##### **ussource** (USHORT)

Source type.

Identifies the type of control:

|                    |  |
|--------------------|--|
| CMDSRC_PUSHBUTTON  | Posted by a push-button control. <i>uscmd</i> is the window identifier of the push button.           |
| CMDSRC_MENU        | Posted by a menu control. <i>uscmd</i> is the identifier of the menu item.                           |
| CMDSRC_ACCELERATOR | Posted as the result of an accelerator. <i>uscmd</i> is the accelerator command value.               |
| CMDSRC_OTHER       | Other source. <i>uscmd</i> gives further control-specific information defined for each control type. |

##### **uspointer** (USHORT)

Pointing-device indicator.

TRUE The message is posted as a result of a pointing-device operation.  
FALSE The message is posted as a result of a keyboard operation.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This message is posted to the queue of the owner of the control, thereby offering it the opportunity to perform some activity as a result.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_SYSCONFIG

This message is posted to all main windows when one of the settable system values is changed.

## Parameters

**param1**

**usChangedFirst** (USHORT)

First system value.

The first of a contiguous set of system values that has been changed.

**param2**

**usChangedLast** (USHORT)

Last system value.

The last of a contiguous set of system values that has been changed.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

If *usChangedFirst* equals *usChangedLast*, only one system value has changed.

If an application changes the settable system values, it is the responsibility of the application to post this message to all main windows.

This message is processed by WC\_FRAME windows by doing any frame-specific processing (such as sending WM\_SETBORDERSIZE messages to the size border if SV\_CX/CYSIZEBORDER system values have changed) and then sending the message to the client window if one exists.

This message is only posted when settable system values change.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_TEXTEDIT

This message occurs when the operator requests a direct name edit operation.

### Parameters

**param1**

**usPointer** (USHORT)

Input device flag.

TRUE     Message resulted from pointer event

FALSE    Message resulted from keyboard event.

**param2**

**ptspointerpos** (POINTS)

Pointer position.

The pointer position is in window coordinates relative to the bottom-left corner of the window. This value is ignored if *usPointer* is not set to TRUE.

### Returns

**rc** (BOOL)

Processed indicator.

TRUE     Message processed

FALSE    Message ignored.

### Remarks

This message is posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information. This message will result from either a mouse event, specified by the system value *SV\_TEXTEDIT*, or a keyboard event, specified by the system value *SV\_TEXTEDITKB*.

### Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message, other than to set result to FALSE.

---

## WM\_TIMER

This message is posted when a timer times out.

### Parameters

#### param1

##### **idTimer** (USHORT)

Timer identity.

Any timer ids that are not being used must be passed on the default window procedure.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is always queued and is processed specially by the `WinGetMsg` and `WinPeekMsg` calls, as follows:

1. Timers are processed only by the `WinGetMsg` and `WinPeekMsg` calls.
2. A timer posts only one `WM_TIMER` message at a time.
3. `WM_TIMER` messages are queued lower than all other messages except `WM_SEM4` messages.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_TRACKFRAME

This message is sent to a window whenever it is to be moved or sized.

### Parameters

**param1**

**fsTrackFlags** (USHORT)

Tracking flags.

Contains a combination of one or more TF\_\* flags; for details, see the TRACKINFO data structure description.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator

TRUE The operation is successful.

FALSE The operation is unsuccessful, or the operation is terminated.

### Remarks

Respond to this message by causing a tracking rectangle to be drawn to move or size the window. For information, see WinTrackRect.

### Default Processing

None.

### Related Messages

- WM\_TRACKFRAME (in Frame Controls)

---

## WM\_TRANSLATEACCEL

This message is sent to the focus window whenever a WM\_CHAR message occurs.

### Parameters

**param1**

**pqmsg** (PQMSG)

Pointer to a QMSG structure.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

**rc (BOOL)**

Translated indicator.

**TRUE** The character exists in the accelerator table and has been translated in the QMSG structure.

**FALSE** The character does not exist in the accelerator table or the window does not have an accelerator table.

## Remarks

Normally, this message is not processed by the focus window, but is passed to its parent, which passes it to its parent, until a frame window is reached.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_TRANSLATEACCEL (in Frame Controls)

---

## WM\_TRANSLATEMNEMONIC

This message occurs during frame control processing of a WM\_TRANSLATEACCEL message.

## Parameters

**param1**

**pqmsg (PQMSG)**

Pointer to a QMSG structure. QMSG structure.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE The character has been translated into an accelerator.

FALSE The character has not been translated into an accelerator.

## Remarks

This message is sent by the frame control to itself during the processing of a WM\_TRANSLATEACCEL message, if the frame control does not translate a character into an accelerator by use of the frame window or queue accelerator tables.

When the frame control receives this message, it sends it to the application menu window, that is the window with identity FID\_MENU.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_TRANSLATEMNEMONIC (in Frame Controls)

---

## WM\_UPDATEFRAME

This message is sent by an application after frame controls have been added or removed from the window frame.

## Parameters

**param1**

**flCreateFlags** (ULONG)

Frame-creation flags.

Contains the FCF\_\* flags that indicate which frame controls have been added or removed.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Processed indicator.

TRUE Message processed

FALSE Message ignored.



## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_UPDATEFRAME (in Frame Controls)

---

## WM\_VRNDISABLED

This message indicates that the window is being sized, or that a WinLockWindowUpdate has been issued for the window or one of its parent windows. Direct drawing to the window should be suspended.

## Parameters

**param1**

**mp1 (VOID)**  
Reserved value.

**param2**

**mp2 (VOID)**  
Reserved value.

## Returns

**returns**

**ulReserved (ULONG)**  
Reserved value, should be 0.

## Remarks

The window procedure is expected to suspend direct drawing to the window.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_VRNENABLED

This message tells a window that its visible region is now unlocked and is valid for drawing on. It also contains a message parameter to inform the window if the visible region was changed.

## Parameters

### param1

#### **ffVisRgnChanged** (BOOL)

Flag indicating whether the visible region has been altered.

TRUE The visible region has been altered. The application needs to query the new visible region.

FALSE The visible region has not been changed.

### param2

#### **mp2** (VOID)

Reserved value.

## Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The visible region, in window coordinates, has been sized, moved or unlocked and drawing can now resume. The *ffVisRgnChanged* parameter is TRUE if the visible region was altered, telling the application whether it needs to recheck the visible area of the window. Direct drawing to the window can be resumed.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_VSCROLL

This message occurs when a vertical scroll-bar control has a significant event to notify to its owner.

## Parameters

### param1

#### **usidentifier** (USHORT)

Scroll bar-control window identifier.

## param2

### sslider (SHORT)

Slider position.

0 Either the operator is not moving the slider with the pointer device, or for the instance when *uscmd* is *SB\_SLIDERPOSITION* the pointer is outside the tracking rectangle when the button is released.

Other Slider position.

### uscmd (USHORT)

Command.

|                          |   |
|--------------------------|---|
| <i>SB_LINEUP</i>         | Sent if the operator clicks on the up arrow of the scroll bar, or presses the <i>VK_UP</i> key.                             |
| <i>SB_LINEDOWN</i>       | Sent if the operator clicks on the down arrow of the scroll bar, or presses the <i>VK_DOWN</i> key.                         |
| <i>SB_PAGEUP</i>         | Sent if the operator clicks on the area above the slider, or presses the <i>VK_PAGEUP</i> key.                              |
| <i>SB_PAGEDOWN</i>       | Sent if the operator clicks on the area below the slider, or presses the <i>VK_PAGEDOWN</i> key.                            |
| <i>SB_SLIDERPOSITION</i> | Sent to indicate the final position of the slider.  |
| <i>SB_SLIDERTRACK</i>    | If the operator moves the scroll bar slider with the pointer device, this is sent every time the slider position changes.   |
| <i>SB_ENDSCROLL</i>      | Sent when the operator has finished scrolling, but only if the operator has not been doing any absolute slider positioning. |

## Returns

### *ulReserved* (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- *WM\_VSCROLL* (in Vertical Scroll Bars)

---

## WM\_WINDOWPOSCHANGED

If this message has any of the values of the *fl* parameter of the SWP structure set, with the exception of the SWP\_NOADJUST and SWP\_NOREDRAW values, it is sent to the window procedure of the window whose position is changed.

This message is also sent if the return value from the WM\_ADJUSTWINDOWPOS is not NULL.

### Parameters

#### param1

##### pswp (PSWP)

SWP structures.

This points to two SWP structures. The first SWP structure describes the entire new window state, whereas the second structure describes the entire old window state. The *fl* parameter of the first structure contains only those indicators corresponding to the state changes that occurred.

#### param2

##### flAwp (ULONG)

Adjust window position status indicators.

The AWF\_\* flags specify the state change of the frame window.

The return value from the WM\_ADJUSTWINDOWPOS message:

- 0 The SWP\_NOADJUST option has been specified.
- Other Adjust window position status indicators.

### Returns

#### ulReserved (ULONG)

Reserved value, should be 0.

### Default Processing

The default window procedure sets *ulReserved* to 0 and sends the following messages, based on the values of the *fl* parameter of the first SWP data structure:

- SWP\_SIZE A WM\_SIZE with the new window size from the first SWP structure
- SWP\_HIDE A WM\_SHOW to hide the new window
- SWP\_SHOW A WM\_SHOW to show the new window.

## Examples

This example processes the WM\_WINDOWPOSCHANGED message and assigns the two structures to pointers.

```
PSWP pswpNew, pswpOld;  
  
case WM_WINDOWPOSCHANGED:  
    pswpNew = PVOIDFROMMP(mp1);  
    pswpOld = pswpNew + 1;
```

---

## Default Dialog Processing

This section describes how messages are processed by the default dialog procedure. The default dialog procedure can be called using `WinDefDlgProc`. A user dialog procedure should make this call for all messages that it does not want to process.

For `WM_*` messages other than those specified in this section the Default Dialog Procedure takes the same action and sets **result** to the same value as in Chapter 13, “Frame Control Window Processing.” In the instance of messages that would be sent to `FID_CLIENT`, they are passed to the default window procedure.

For any other messages the default window procedure takes no action, other than to set **reply** to `NULL`.

---

## WM\_CHAR (Default Dialogs)

For the cause of this message, see “`WM_CHAR`” on page 10-32.

For a description of the parameters, see “`WM_CHAR`” on page 10-32.

### Default Processing

If `KC_CHAR` is the mnemonic for a button that already has the focus, a `BM_CLICK` is sent to that button and `rc` is set to `TRUE`. If the button does not have the focus, it receives the focus and `rc` is set to `TRUE`.

If `usvk` contains the value `VK_TAB`, the focus is set to the next tab item in the dialog. `rc` is set to `TRUE`.

If `usvk` contains the value `VK_BACKTAB`, the focus is set to the previous tab item in the dialog. `rc` is set to `TRUE`.

If `usvk` contains the value `VK_LEFT` or `VK_UP`, the focus is set to the previous item in the group. `rc` is set to `TRUE`.

If `usvk` contains the value `VK_RIGHT` or `VK_BOTTOM`, the focus is set to the next item in the group. `rc` is set to `TRUE`.

If `usvk` contains the value `VK_ENTER` or `VK_NEWLINE`, and a push button has the focus, a `BM_CLICK` is sent to the button and `rc` is set to `TRUE`. If another control in the dialog has the focus the dialog is searched for a push button with style `BS_DEFAULT`. If a push button of this style is found, a `BM_CLICK` is sent to that button and `rc` is set to `TRUE`.

If `usvk` contains the value `VK_ESC`, `WM_COMMAND` is posted, with `ussource` is set to `CMDSRC_PUSHBUTTON` and `uscmd` is set to `DID_CANCEL`. `rc` is set to `TRUE`.

In other instances, if an owner exists the message is sent to the owner, otherwise `rc` is set to `FALSE`.

## Related Messages

- WM\_CHAR

---

## WM\_CLOSE (Default Dialogs)

For the cause of this message, see “WM\_CLOSE” on page 10-35.

For a description of the parameters, see “WM\_CLOSE” on page 10-35.

### Default Processing

The default dialog procedure responds to this message by dismissing the dialog by issuing the `WinDismissDlg` function with its *rc* parameter set to `DID_CANCEL`.

## Related Messages

- WM\_CLOSE

---

## WM\_COMMAND (Default Dialogs)

For the cause of this message, see `WM_COMMAND`.

For a description of the parameters, see `WM_COMMAND`.

### Default Processing

The default dialog procedure responds to this message by dismissing the dialog and passing *uscmd* (the control item identifier) as *ulReply* of the `WinProcessDlg` or the `WinDlgBox` function that initiated the dialog. It sets *ulReserved* to 0.

## Related Messages

- WM\_COMMAND

---

## WM\_INITDLG (Default Dialogs)

For the cause of this message, see “WM\_INITDLG” on page 10-52.

For a description of the parameters, see “WM\_INITDLG” on page 10-52.

### Remarks

This message is sent to the dialog procedure, before the dialog box is shown, thereby offering the dialog procedure the opportunity to perform the initialization of the dialog box.

If any string substitutions are made by the `WinSubstituteStrings` call when the dialog is created, the `WM_SUBSTITUTESTRING` message may have been sent before the `WM_INITDLG` message is sent.

### Default Processing

The default dialog procedure passes this message to the default window procedure, which sets *rc* to `FALSE`.

## Related Messages

- WM\_INITDLG

---

## WM\_MATCHMNEMONIC (Default Dialogs)

For the cause of this message, see “WM\_MATCHMNEMONIC” on page 10-55.

For a description of the parameters, see “WM\_MATCHMNEMONIC” on page 10-55.

### Remarks

This message is only processed by Button and Static Controls; all other controls return FALSE.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_MATCHMNEMONIC

---

## WM\_QUERYDLGCODE

This message is sent by the dialog manager to identify the type of control, to determine what kinds of messages the control understands, and also to determine whether an input message may be processed by the dialog manager or passed down to the control.

### Parameters

**param1**

**pQmsg** (PQMSG)

Message queue structure.

This points to a QMSG structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulDialogCode** (ULONG)

Dialog code information flags.

**DLGC\_ENTRYFIELD** Identifies an entry field control. Assumed to understand the EM\_SETSEL message.



|                  |   |
|------------------|---|
| DLGC_BUTTON      | Identifies a button item. Assumed to understand the BM_CLICK message.   |
| DLGC_RADIOBUTTON | Identifies a radio button control. Used with the DLGC_BUTTON code.  |
| DLGC_STATIC      | Identifies a static control. Static controls are not included in arrow key enumeration.   |
| DLGC_DEFAULT     | Identifies a default push-button control.   |
| DLGC_PUSHBUTTON  | Identifies a nondefault push button.  |
| DLGC_CHECKBOX    | Identifies a check-box item. Used with the DLGC_BUTTON code.  |
| DLGC_SCROLLBAR   | Identifies a scroll bar control.  |
| DLGC_MENU        | Identifies a menu control.  |
| DLGC_TABONCLICK  | Used by static controls to indicate that a mouse click on this control will cause focus to be placed on the next control in the dialog that has the WP_TABSTOP style. This should be used in combination with the DLGC_STATIC code. |
| DLGC_MLE         | Identifies a multiline entry field control.   |

### Remarks

When processing user input, the dialog manager makes some assumptions about the operation of specific controls. The dialog manager sends the WM\_QUERYDLGCODE message to obtain a code that governs what assumptions can be made.

If the window receiving this message is *not* a control as defined above, this message returns 0.

### Default Processing

The default dialog procedure takes no action on this message, other than to set *uiDialogCode* to NULL.

---

## Default File Dialog Processing

This section describes how messages are processed by the default dialog procedure of the file dialog. This standard dialog can be used to provide a common, consistent file selection function.

The file dialog's default procedure can be called using the WinDefFileDlgProc function. A user-provided subclassing dialog procedure should make this call for all messages that it does not process when using the file dialog.

The default dialog procedure of the file dialog sends the messages listed in this section to itself to perform the requested action. This design allows a user-provided dialog procedure to customize the file dialog to its own needs.

---

## **FDM\_ERROR**

This message is sent whenever the file dialog is going to display an error message window. This allows an application to display its own message, if desired, instead of messages provided by the system.

### **Parameters**

#### **param1**

##### **usErrorId** (USHORT)

Error message ID.

This is the ID of the message that is displayed by the file dialog if the default file dialog procedure processes the message.

#### **param2**

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### **Returns**

#### **usUserReply** (USHORT)

User's reply.

|             |   |
|-------------|---|
| 0           | The file dialog presents the error message for this ID.   |
| MBID_OK     | The file dialog processes the reply as if the OK push button was pressed in its message window.     |
| MBID_CANCEL | The file dialog processes the reply as if the Cancel push button was pressed in its message window. |
| MBID_RETRY  | The file dialog processes the reply as if the Retry push button was pressed in its message window.  |

### **Remarks**

The application uses this message to provide application-specific error messages in response to file dialog errors that are detected during file dialog processing. The application can choose whether to allow the dialog to present its message or whether to provide its own message and return the response from that message window to the dialog for processing.

### **Default Processing**

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return NULL.

---

## FDM\_FILTER

This message is sent before a file that meets the current filter criteria is added to the File list box.

### Parameters

**param1**

**pFilename** (PSZ)

Pointer to the file name.

**param2**

**pEAType** (PSZ)

Pointer to the .TYPE EA extended attribute.

### Returns

**rc** (BOOL)

Success indicator.

TRUE     Add the file.

FALSE    Do not add the file.

### Remarks

The application checks this message to obtain the name and the .TYPE EA extended attribute of the file to be added. The application then determines whether or not the file will be added.

When FALSE is returned, the file is not added to the dialog's list box.

### Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return FALSE.

---

## FDM\_VALIDATE

This message is sent when the user selects a file and presses Enter or clicks on the OK button, or double-clicks on a file name in the file list box.

### Parameters

**param1**

**pFileName** (PSZ)

Pointer to the fully-qualified file name.

## param2

### **usSeltype** (USHORT)

Selection type.

### **rc** (BOOL)

Validity indicator.

TRUE File name is valid.

FALSE File name is not valid.

## Remarks

This message is only sent just before the dialog returns to the caller with the user-selected file name. Before this message is sent, *pFileName* is updated with the user-selected file name. The application can determine if this file name is acceptable. For instance, if the file dialog is being used to pick a "SaveAs" file name, the application can check to see if the file is read-only. If it is, a warning dialog should be brought up to notify the user.

When FALSE is returned from a FDM\_VALIDATE message, the dialog will not be dismissed and the user can continue to use the File Dialog to select an alternate file.

In multiple file selection dialogs this message is sent for each selected entry within the file list box. When the name of the file being validated comes from a selected entry in the list box, *param2* will contain FDS\_LBSELECTION. When the name of the file comes from the file name entry field, *param2* will contain FDS\_EFSELECTION. Single file selection dialogs will always return FDS\_EFSELECTION in *param2* since the returned file name always comes from the single line entry field.

## Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return FALSE.

---

## Default Font Dialog Processing

This section describes how messages are processed by the default dialog procedure of the font dialog. This standard dialog can be used to provide a common, consistent font selection function.

The font dialog's default procedure can be called using the WinDefFontDlgProc function. A user-provided subclassing dialog procedure should make this call for all messages that it does not process when using the font dialog.

The default dialog procedure of the font dialog sends the messages listed in this section to itself to perform the requested action. This design allows a user-provided dialog procedure to customize the font dialog to its own needs.

---

## WM\_DRAWITEM (in Font Dialog)

If the FNTS\_OWNERDRAWPREVIEW style is set for a font dialog, this notification message is sent to that dialog's owner whenever the preview window area (sample text) is to be drawn.

### Parameters

#### param1

##### **id (USHORT)**

Window identifier.

The window ID of the sample area (DID\_SAMPLE).

#### param2

##### **pOwnerItem (POWNERITEM)**

Pointer to an OWNERITEM data structure.

The following list defines the OWNERITEM data structure fields as they apply to the font dialog. See "OWNERITEM" on page A-136 for the default field values.

##### **hwnd (HWND)**

Window handle of the sample area.

##### **hps (HPS)**

Presentation-space handle.

##### **fsState (ULONG)**

Reserved.

##### **fsAttribute (ULONG)**

Reserved.

##### **fsStateOld (ULONG)**

Reserved.

##### **fsAttributeOld (ULONG)**

Reserved.

##### **rclItem (RECT)**

Item rectangle to be drawn in window coordinates.

##### **idlItem (LONG)**

Reserved.

##### **hlItem (CNRDRAWITEMINFO)**

Reserved.

## Returns

**rc** (BOOL)

Item-drawn indicator.

**TRUE** The owner draws the item.

**FALSE** If the owner does not draw the item, the owner returns this value and the font dialog draws the item.

## Remarks

The font dialog provides this message to give the application the opportunity to provide a custom drawn preview area.

The font dialog default dialog procedure generates this message and sends it to its owner, informing the owner that the preview area is to be drawn. The owner is then given the opportunity to draw that area and to indicate that the area has been drawn or that the font dialog is to draw it.

## Default Processing

For a description of the default processing, see `WM_DRAWITEM`.

---

## FNTM\_FACENAMECHANGED

This message notifies the subclassing application whenever the font family name is changed by the user.

## Parameters

**param1**

**pFamilyname** (PSZ)

Pointer to the currently-selected face name.

**param2**

**uiReserved** (ULONG)

Reserved value, should be 0.

## Returns

**uiReserved** (ULONG)

Reserved value, should be 0.

## Remarks

*pFamilyname* is the currently selected family name. The application can modify this string if it desires. The buffer set aside is the maximum size a face name string can be (`FACESIZE`).

## Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return 0.

---

## FNTM\_FILTERLIST

This message is sent whenever the Font Dialog is preparing to add a font family name, font style type, or point size entry to the combination box fields that contain these parameters.

### Parameters

#### param1

##### pFontname (PSZ)

Pointer to the text string that is being added to the combination box.

#### param2

##### usFieldId (USHORT)

Field identifier.

The identifier of the field to which the text string is being added. The identifier can be one of the following:

- |                 |  |
|-----------------|--|
| FNTI_FAMILYNAME | The text string is an addition to the family name combination box. |
| FNTI_STYLENAME  | The text string is an addition to the style combination box.       |
| FNTI_POINTSIZE  | The text string is an addition to the size combination box.        |

##### usFontType (USHORT)

Font information.

The family name, style, or point size that is being added to the combination box. Use one of the following to identify the font information that is being added:

- |                       |   |
|-----------------------|---|
| FNTI_BITMAPFONT       | A bit-map font is being added or a point size of a bit-map font is being added.       |
| FNTI_VECTORFONT       | A vector font is being added.   |
| FNTI_SYNTHESIZED      | A synthesized font is being added. This value is valid for the style field only.      |
| FNTI_FIXEDWIDTHFONT   | A fixed width (monospace) font is being added.  |
| FNTI_PROPORTIONALFONT | A proportionally spaced font is being added.  |
| FNTI_DEFAULTLIST      | A point size from the default list (or the application-supplied list) is being added. |

## Returns

**rc** (BOOL)

Filter indicator.

TRUE     Add the text string to the combination box.

FALSE    Do not add the text string to the combination box.

## Remarks

The application checks this message to obtain the name and the .TYPE EA extended attribute of the file being added. The application then determines whether or not the file will be added.

When FALSE is returned, the file is not added to the dialog's list box.

## Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return FALSE.

---

## FNTM\_POINTSIZECHANGED

This message notifies subclassing applications when the point size of the font is changed by the user.

## Parameters

**param1**

**pPointSize** (PSZ)

Pointer to the text in the point-size entry field.

**param2**

**fxPointSize** (FIXED)

Point size.

The *fxPointSize* field in FONTDLG stated in fixed-point notation.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

When the application wants to limit the point sizes the user can select, it should process this message by changing the *pPointSize* value and putting up a message box explaining the limitation to the user.



## Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return 0.

---

## FNTM\_STYLECHANGED

This message notifies subclassing applications when the user changes any of the attributes in the STYLECHANGE structure.

### Parameters

#### param1

**styc** (STYLECHANGE)  
Style changes.

#### param2

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Remarks

The "Old" fields show the style attributes before the user made the change. The other parameters show what the state will be after the application passes this message to WinDefFontDlgProc. When the "Old" field and the "New" field are the same, no change is made for that attribute.

### Default Processing

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return 0.

---

## **FNTM\_UPDATEPREVIEW**

This message notifies subclassing applications before the preview window is updated. This occurs when the font selection is modified.

### **Parameters**

#### **param1**

##### **hwndPreview (HWND)**

Window handle.

Window handle the preview image is drawn into. This is a static text field.

#### **param2**

##### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Remarks**

This message notifies an application that the dialog is about to update the preview area.

### **Default Processing**

The WinDefDlgProc function does not expect to receive this message and takes no action on it other than to return 0.

---

## Language Support Window Processing

This system-provided window procedure processes messages for a window that has been created with a window class specifying a "NULL" window procedure.

The following describes the WM\_\* messages and the language support window procedure action.

For any other messages the Language Support Window Procedure performs the same actions as the Default Window Procedure.

---

### WM\_ACTIVATE (Language Support Window)

For the cause of this message, see "WM\_ACTIVATE" on page 10-5.

For a description of the parameters, see "WM\_ACTIVATE" on page 10-5.

#### Remarks

The Language Support Window Procedure responds to this message by posting a WM\_PACTIVATE message to the application queue and setting *ulReserved* to 0.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_ACTIVATE

---

### WM\_CONTROL (Language Support Window)

For the cause of this message, see WM\_CONTROL.

For a description of the parameters, see WM\_CONTROL.

#### Remarks

The Language Support Window Procedure responds to this message by posting a WM\_PCONTROL message to the application queue and setting *ulReserved* to 0.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_CONTROL

---

## **WM\_PAINT (Language Support Window)**

For the cause of this message, see “WM\_PAINT” on page 10-66.

For a description of the parameters, see “WM\_PAINT” on page 10-66.

### **Remarks**

The Language Support Window Procedure responds to this message by posting a WM\_PPAINT message to the application queue and setting *ulReserved* to 0.

The WinBeginPaint and WinEndPaint functions are issued by the Language Support Window Procedure, during the processing of the WM\_PPAINT message.

### **Default Processing**

The default window procedure issues the WinBeginPaint and WinEndPaint functions, and then sets *ulReserved* to 0.

### **Related Messages**

- WM\_PAINT

---

## **WM\_PPAINT (Language Support Window)**

For the cause of this message, see “WM\_PPAINT” on page 10-68.

For a description of the parameters, see “WM\_PPAINT” on page 10-68.

### **Remarks**

The Language Support Window Procedure issues the WinBeginPaint and WinEndPaint functions, and then sets *ulReserved* to 0.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_PPAINT

---

## **WM\_SETFOCUS (Language Support Window)**

For the cause of this message, see "WM\_SETFOCUS" on page 10-83.

For a description of the parameters, see "WM\_SETFOCUS" on page 10-83.

### **Remarks**

The Language Support Window Procedure responds to this message by posting a WM\_PSETFOCUS message to the application queue and setting *ulReserved* to 0.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_SETFOCUS

---

## **WM\_SIZE (Language Support Window)**

For the cause of this message, see WM\_SIZE.

For a description of the parameters, see WM\_SIZE.

### **Remarks**

The Language Support Window Procedure responds to this message by posting a WM\_PSIZE message to the application queue and setting *ulReserved* to 0.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_SIZE

---

## **WM\_SYSCOLORCHANGE (Language Support Window)**

For the cause of this message, see “WM\_SYSCOLORCHANGE” on page 10-90.

For a description of the parameters, see “WM\_SYSCOLORCHANGE” on page 10-90.

### **Remarks**

The Language Support Window Procedure responds to this message by posting a WM\_PSYSCOLORCHANGE message to the application queue and setting *ulReserved* to 0.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_SYSCOLORCHANGE

---

## Language Support Dialog Processing

This system-provided window procedure processes messages for a dialog that has been created or loaded specifying a 'NULL' dialog procedure.

For any other messages the Language Support Dialog Procedure issues and returns the result of the `WinDefDlgProc` function.

---

### WM\_ACTIVATE (Language Support Dialog)

For the cause of this message, see "WM\_ACTIVATE" on page 10-5.

For a description of the parameters, see "WM\_ACTIVATE" on page 10-5.

#### Remarks

The Language Support Dialog Procedure responds to this message by issuing the `WinDefDlgProc` function, then posting a `WM_PACTIVATE` message to the application queue and setting *ulReserved* to the result of the `WinDefDlgProc` function.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_ACTIVATE

---

### WM\_CONTROL (Language Support Dialog)

For the cause of this message, see "WM\_CONTROL" on page 10-39.

For a description of the parameters, see "WM\_CONTROL" on page 10-39.

#### Remarks

The Language Support Dialog Procedure responds to this message by issuing the `WinDefDlgProc` function, then posting a `WM_PCONTROL` message to the application queue and setting *ulReserved* to the result of the `WinDefDlgProc` function.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_CONTROL

---

## **WM\_PAINT (Language Support Dialog)**

For the cause of this message, see “WM\_PAINT” on page 10-66.

For a description of the parameters, see “WM\_PAINT” on page 10-66.

### **Remarks**

The Language Support Dialog Procedure responds to this message by issuing the WinDefDlgProc function, then posting a WM\_PPAINT message to the application queue and setting *ulReserved* to the result of the WinDefDlgProc function.

The WinBeginPaint and WinEndPaint functions are issued by the Language Support Dialog Procedure, during the processing of the WM\_PPAINT message.

### **Default Processing**

The default window procedure issues the WinBeginPaint and WinEndPaint functions, and then sets *ulReserved* to 0.

### **Related Messages**

- WM\_PAINT

---

## **WM\_PPAINT (Language Support Dialog)**

For the cause of this message, see “WM\_PPAINT” on page 10-68.

For a description of the parameters, see “WM\_PPAINT” on page 10-68.

### **Remarks**

The Language Support Dialog Procedure issuing the WinDefDlgProc function, then issues the WinBeginPaint and WinEndPaint functions, and then setting *ulReserved* to the result of the WinDefDlgProc function.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_PPAINT



---

## **WM\_SETFOCUS (Language Support Dialog)**

For the cause of this message, see “WM\_SETFOCUS” on page 10-83.

For a description of the parameters, see “WM\_SETFOCUS” on page 10-83.

### **Remarks**

The Language Support Dialog Procedure responds to this message by issuing the WinDefDlgProc function, then posting a WM\_PSETFOCUS message to the application queue and setting *ulReserved* to the result of the WinDefDlgProc function.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_SETFOCUS

---

## **WM\_SIZE (Language Support Dialog)**

For the cause of this message, see “WM\_SIZE” on page 10-88.

For a description of the parameters, see “WM\_SIZE” on page 10-88.

### **Remarks**

The Language Support Dialog Procedure responds to this message by issuing the WinDefDlgProc function, then posting a WM\_PSIZE message to the application queue and setting *ulReserved* to the result of the WinDefDlgProc function.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### **Related Messages**

- WM\_SIZE

---

## **WM\_SYSCOLORCHANGE (Language Support Dialog)**

For the cause of this message, see “WM\_SYSCOLORCHANGE” on page 10-90.

For a description of the parameters, see “WM\_SYSCOLORCHANGE” on page 10-90.

### **Remarks**

The Language Support Dialog Procedure responds to this message by issuing the WinDefDlgProc function, then posting a WM\_PSYSCOLORCHANGE message to the application queue and setting *ulReserved* to the result of the WinDefDlgProc function.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## **Related Messages**

- WM\_SYSCOLORCHANGE



---

## Chapter 11. Button Control Window Processing

This system-provided window procedure processes the actions on a button control (WC\_BUTTON).

### Purpose

A button control is a small rectangular child window representing a button that the operator can “switch” on or off. Button controls can be used alone or in groups, and can either be labeled or appear without text. Button controls typically change appearance when the operator clicks a pointing device on them or pressing the space bar when the button has the keyboard focus.

Buttons can be disabled to prevent them from responding when the operator clicks on them. Disabled buttons are displayed using a different emphasis technique (for example, color or half-toning).

---

### Button Control Styles

These button control styles are available:

- |                           |  |
|---------------------------|--|
| <b>BS_AUTOCHECKBOX</b>    | An automatic check box automatically toggles its state whenever the user clicks on it.   |
| <b>BS_AUTORADIOBUTTON</b> | When clicked, an automatic radio button automatically checks itself and unchecks all other radio buttons in the same group.  |
| <b>BS_AUTOSIZE</b>        | Buttons with this style are sized automatically to make sure the contents fit.<br><br>If BS_AUTOSIZE is selected when the button is created, and a -1 is specified for either the cx or cy parameter of WinCreateWindow, (or when creating the button as a resource) then the button's optimal size is calculated to display the its contents. |
| <b>BS_AUTO3STATE</b>      | An automatic three-state check box automatically toggles its state when the user clicks on it.   |
| <b>BS_BITMAP</b>          | Places a bit map instead of text on the push button control. This style works only with the BS_PUSHBUTTON.   |
| <b>BS_CHECKBOX</b>        | A check box is a small square with a character string to the right. If it is checked, a small black box appears inside the small square. When the box or string is clicked, by clicking on it with the pointing device or pressing the keyboard spacebar when it is active,  |
| <b>BS_DEFAULT</b>         | A BS_DEFAULT pushbutton is one with a thick border box. It has the same properties as a pushbutton. In addition, the user may press a BS_DEFAULT pushbutton by pressing the RETURN or ENTER key. The intention is the same for   |

user-buttons, but the appearance of a BS\_DEFAULT userbutton is application defined.

This style can be ORed with the BS\_PUSHBUTTON and BS\_USERBUTTON styles:

**BS\_HELP**

The button posts a WM\_HELP message rather than a WM\_COMMAND message.

This style can be ORed with the BS\_PUSHBUTTON style.

If both BS\_HELP and BS\_SYSCOMMAND are set, BS\_HELP takes precedence.

**BS\_ICON**

Places an icon instead of text on the push button control. This style works only with the BS\_PUSHBUTTON style.

**BS\_MINIICON**

This enables miniicons (half the size of normal icons) to be placed on the push button control.

**BS\_NOBORDER**

The pushbutton is displayed without a border drawn around it. There is no other change in the pushbutton's operation.

This style can be ORed with the BS\_PUSHBUTTON style.

**BS\_NOCURSORSELECT**

The radio button does not select itself when given the focus as the result of an arrow key or tab key.

This style can be ORed with the BS\_AUTORADIOBUTTON style.

**BS\_NOPOINTERFOCUS**

Buttons with this style do not set the focus to themselves when clicked with the pointing device. This enables the cursor to stay on a control for which information is required, rather than moving to the button. This style has no effect on keyboard interaction. The tab key can still be used as usual to move the focus to the button.

This style can be ORed with any of the basic button styles.

**BS\_PUSHBUTTON**

A pushbutton is a box that contains a string. When a button is pushed, by clicking the pointing device on it or pressing the spacebar when it is active, the parent window is notified.

**BS\_RADIOBUTTON**

A radio button is similar to a check box, but is typically used in groups in which only one button at a time is checked. When a radio button is clicked or a cursor key is pressed to move within the group, it notifies its owner window. It is then up to the owner window to check the clicked radio button and uncheck all the rest, if necessary.

**BS\_SYSCOMMAND**

The button posts a WM\_SYSCOMMAND message rather than a WM\_COMMAND message.

This style can be ORed with the BS\_PUSHBUTTON style.

If both BS\_HELP and BS\_SYSCOMMAND are set, BS\_HELP takes

|                      |   |
|----------------------|---|
| <b>BS_TEXT</b>       | This enables both text and a bitmap, icon, or miniicon to be placed on the push button control. This style works only with the BS_PUSHBUTTON style, and should be used in conjunction with BS_BITMAP, BS_ICON or BS_MINIICON. |
| <b>BS_USERBUTTON</b> | This is an application-definable button. The owner window of this style control receives the additional button style BN_PAINT.  |
| <b>BS_3STATE</b>     | A three-state check box is identical to a check box control except that its check box can be half-toned as well as the box being checked or unchecked.  |

When BS\_ICON, BS\_MINIICON or BS\_BITMAP is selected, the image can be activated by specifying the image ID with the button text string. For instance, to load an icon (#define ICON\_ID 300), and display it within a button, the button string is set to "#300."

When BS\_ICON, BS\_MINIICON or BS\_BITMAP is selected along with BS\_TEXT, the image can still be activated by specifying the following with a zero-terminated text string. format:

```
"#<image-id>\t<text>"
```

where:

|            |   |
|------------|---|
| <image-id> | resource id of the icon, miniicon or bitmap |
| \t         | tab character                               |
| <text>     | zero-terminated button text string          |

For example, to load an icon (#define ICON\_ID 300) and display it with the button text "My Button," the button string is set to "#300\tMy Button." Notice the "\t" is used to separate the text from the image-id. The image is displayed above the text within the button.

---

## Button Control Data

See "BTNCDATA" on page A-24.

---

## Default Colors

The following system colors are used when the system draws button controls:

- SYSCLR\_BUTTONDEFAULT
- SYSCLR\_BUTTONLIGHT
- SYSCLR\_BUTTONMIDDLE
- SYSCLR\_MENUTEXT
- SYSCLR\_WINDOW
- SYSCLR\_WINDOWFRAME.

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_BACKGROUND  
PP\_BORDER  
PP\_DISABLEDFOREGROUND  
PP\_FOREGROUND  
PP\_HILITEFOREGROUND.

---

## Button Control Notification Messages

These messages are initiated by the button control window to notify its owner of significant events.

---

### WM\_COMMAND (in Button Controls)

For the cause of this message, see “WM\_COMMAND” on page 10-37.

For a description of the parameters, see “WM\_COMMAND” on page 10-37.

Button control sets *uscmd* to the button identity and *ussource* to `CMDSRC_PUSHBUTTON`.

#### Remarks

The button control generates this message when a push button of style `BS_PUSHBUTTON` is pressed or when it receives a `BM_CLICK` message. The button control posts the message to the queue of the control owner.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_COMMAND

---

### WM\_CONTROL (in Button Controls)

For the cause of this message, see “WM\_CONTROL” on page 10-39.

#### Parameters

**param1**

**id** (USHORT)

Button control identity.

**usnotifycode** (USHORT)

Notification code.

The notification code `BN_PAINT` is only generated when the button control has a style of `BS_USERBUTTON`.



The button control uses these notification codes:

|               |   |
|---------------|---|
| BN_CLICKED    | The button has been pressed.  |
| BN_DBLCLICKED | The button has been double-clicked.                                   |
| BN_PAINT      | The button requires painting, using one of the following draw states: |
| BDS_DISABLED  | The disabled state of the button requires painting.                   |
| BDS_HILITED   | The highlighted state of the button requires painting.                |
| BDS_DEFAULT   | The default state of the button requires painting.                    |

## param2

### **fcontrolspect** (ULONG)

Control-specific information.

When *usnotifycode* is BN\_PAINT this parameter is a pointer to a USERBUTTON structure, otherwise this parameter is the window handle of the button control.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The button control generates this message and sends it to its owner, informing the owner of this event, when:

- Its style is not BS\_PUSHBUTTON and the button is pressed.
- It receives a BM\_CLICK message.
- Its style is BS\_USERBUTTON and the button is clicked or double clicked.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_CONTROL

---

## WM\_HELP (in Button Controls)

For the cause of this message, see “WM\_HELP” on page 10-49.

For a description of the parameters, see “WM\_HELP” on page 10-49.

Button control sets *uscmd* to the button identity.

### Remarks

This message is identical to a WM\_COMMAND message, but implies that the application should respond to this message by displaying help information.

The button control generates this message and posts it to the queue of its owner, if it has the style of BS\_HELP and a push button is pressed, or when it receives a BM\_CLICK message.

### Default Processing

The default window procedure sends this message to the parent window, if it exists and is not the desktop. Otherwise, it sets *ulReserved* to 0.

### Related Messages

- WM\_HELP

---

## WM\_SYSCOMMAND

For the cause of this message, see “WM\_SYSCOMMAND” on page 10-91.

For a description of the parameters, see “WM\_SYSCOMMAND” on page 10-91.

Button control sets *uscmd* to the button identity.

### Remarks

If the button control is specified with a style of BS\_SYSCOMMAND but not with BS\_HELP, the button control generates this message and posts it to the queue of its owner when a push button is pressed, or when it receives a BM\_CLICK message.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## Button Control Window Messages

This section describes the Button Control Window Procedure actions on receiving the following messages.

---

### BM\_CLICK

An application sends this message to cause the effect of the operator clicking a push button.

#### Parameters

##### param1

###### usUp (USHORT)

Up and down indicator.

TRUE     Perform the default upclick action  
FALSE    Perform the default downclick action.

##### param2

###### ulReserved (ULONG)

Reserved value, should be 0.

#### Returns

##### ulReserved (ULONG)

Reserved value, should be 0.

#### Remarks

The button control responds to this message by taking the action that occurs if the button is clicked by the operator. This causes the following messages to be generated:

- A WM\_HELP (in Button Controls) message, if the button has a style of BS\_HELP.
- A WM\_SYSCOMMAND message, if the button has a style of BS\_PUSHBUTTON and a style of BS\_SYSCOMMAND and not a style of BS\_HELP.
- A WM\_COMMAND (in Button Controls) message, if the button has a style of BS\_PUSHBUTTON but not a style of BS\_SYSCOMMAND and not a style of BS\_HELP.
- A WM\_CONTROL (in Button Controls) message, whose *usnotifycode* is set to BN\_CLICKED, if the button has a style of BS\_USERBUTTON, BS\_PUSHBUTTON, BS\_CHECKBOX, or BS\_3STATE, and not a style of BS\_SYSCOMMAND or BS\_HELP.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ulReserved* to the default value of 0.

---

## BM\_QUERYCHECK

This message returns the checked state of a button control.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usCheck** (USHORT)

Check indicator.

- 0 The button control is in unchecked state.
- 1 The button control is in checked state.
- 2 The button control is in indeterminate state.

### Remarks

The button control responds to this message, if it has a style of BS\_CHECKBOX, BS\_AUTOCHECKBOX, BS\_RADIOBUTTON, BS\_AUTORADIOBUTTON, BS\_3STATE, or BS\_AUTO3STATE, by setting *usCheck* as appropriate.

If the button has any other style, the button control takes no action other than to set *usCheck* to 0.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *usCheck* to the default value of 0.

---

## BM\_QUERYCHECKINDEX

This message returns the zero-based index of a checked radio button.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**sindex** (SHORT)

Radio-button index.

-1 No radio button of the group is checked, or this button control does not have the style BS\_RADIOBUTTON or BS\_AUTORADIOBUTTON.

Other Zero-based index of the checked radio button of the group.

### Remarks

The button control responds to this message by setting *sindex* as appropriate.

This message may be sent to any radio button or autoradio button in a group of buttons. For details of the WS\_GROUP style, see "Window Styles" on page 10-3.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sindex* to the default value of 0.

---

## BM\_QUERYHILITE

This message returns the highlighting state of a button control.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Highlight indicator.

TRUE The button control is displayed in highlighted state.

FALSE The button control is displayed in unhighlighted state.

## Remarks

The button control responds to this message, if it has a style of BS\_PUSHBUTTON, by setting *rc* as appropriate.

If the button has any other style, the button control takes no action other than to set *rc* to FALSE.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, except to set *rc* to the default value of FALSE.

---

## BM\_SETCHECK

This message sets the checked state of a button control.

### Parameters

**param1**

**uscheck** (USHORT)

Check state.

- 0 Display the button control in the unchecked state
- 1 Display the button control in the checked state
- 2 Display a 3-state button control in the indeterminate state.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usoldstate** (USHORT)

Old check state of the button control.

- 0 Unchecked
- 1 Checked
- 2 Indeterminate.

### Remarks

The button control responds to this message by displaying it in the appropriate state and returning the old state.

If the button control has the style of BS\_CHECKBOX, BS\_AUTOCHECKBOX, BS\_RADIOBUTTON, or BS\_AUTORADIOBUTTON, it is displayed in the checked state if *uscheck* is set to 1, or in the unchecked state if it is set to 0 and *usoldstate* is set as appropriate.

If the button control has the style of BS\_RADIOBUTTON or BS\_AUTORADIOBUTTON, the WS\_TABSTOP style is modified. If the resulting state of the button is checked, the WS\_TABSTOP style is set, otherwise it is reset.

If the button control has the style of BS\_3STATE or BS\_AUTO3STATE, it is displayed in the unchecked state if *uscheck* is set to 0, in the checked state if it is set to 1, and in the indeterminate state if it is set to 2 and *usoldstate* is set as appropriate.

If the button control has the style of BS\_USERBUTTON, a WM\_CONTROL (in Button Controls) message is sent to its owner with *usnotifycode* set to BN\_PAINT and *usoldstate* is set as appropriate.

If the button control has any other style, the button control takes no action other than to set *usoldstate* to 0.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, except to set *usoldstate* to the default value of 0.

---

## BM\_SETDEFAULT

This message sets the default state of a button control.

### Parameters

**param1**

**usdefault** (USHORT)

Default state.

TRUE     Display the button control in the default state

FALSE    Display the button control in the nondefault state.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE     Successful operation

FALSE    Error occurred.

## Remarks

The button control responds to this message, if it has a style of BS\_USERBUTTON or BS\_PUSHBUTTON, by displaying the button control in the default or nondefault state as appropriate, and setting *rc* to TRUE.

If the button control has any other style, the button control takes no action other than to set *rc* to FALSE.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## BM\_SETHILITE

This message sets the highlight state of a button control.

## Parameters

**param1**

**ushilite** (USHORT)

Highlight indicator.

TRUE     Display the button control in the highlighted state

FALSE    Display the button control in the unhighlighted state.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**foldstate** (BOOL)

Old highlight state.

TRUE     The button control was in highlighted state

FALSE    The button control was in unhighlighted state.

## Remarks

The button control responds to this message, if it has a style of BS\_PUSHBUTTON, BS\_CHECKBOX, BS\_AUTOCHECKBOX, BS\_RADIOBUTTON, BS\_AUTORADIOBUTTON, BS\_3STATE, or BS\_AUTO3STATE, by displaying the button control in the appropriate highlight state and setting *foldstate* as appropriate.

If the style of the Button Control is BS\_USERBUTTON, a WM\_CONTROL (in Button Controls) message is sent to its owner with *usnotifycode* set to BN\_PAINT and with *fcontrolspect* pointing to a USERBUTTON structure and sets *foldstate* as appropriate.



## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *foldstate* to the default value of FALSE.

---

## WM\_ENABLE (in Button Controls)

For the cause of this message, see “WM\_ENABLE” on page 10-43.

For a description of the parameters, see “WM\_ENABLE” on page 10-43.

### Remarks

This message notifies the button control window procedure of a change in the enable state of the button.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### Related Messages

- WM\_ENABLE

---

## WM\_MATCHMNEMONIC (in Button Controls)

For the cause of this message, see “WM\_MATCHMNEMONIC” on page 10-55.

For a description of the parameters, see “WM\_MATCHMNEMONIC” on page 10-55.

### Remarks

The button control window procedure responds to this message by setting *rc* as appropriate. If MP1 matches the button mnemonic, return *rc* to TRUE.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_MATCHMNEMONIC

---

## **WM\_QUERYCONVERTPOS (in Button Controls)**

For the cause of this message, see “WM\_QUERYCONVERTPOS” on page 10-72.

For a description of the parameters, see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Remarks**

The button control window procedure returns QCP\_NOCONVERT.

### **Default Processing**

For the default window procedure processing of this message see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Related Messages**

- WM\_QUERYCONVERTPOS

---

## **WM\_QUERYWINDOWPARAMS (in Button Controls)**

Occurs when an application queries the button control window procedure window parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### **Remarks**

The button control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to zero and sets *rc* to FALSE.

### **Related Messages**

- WM\_QUERYWINDOWPARAMS

---

## **WM\_SETWINDOWPARAMS (in Button Controls)**

Occurs when an application sets or changes the button control window procedure window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### **Remarks**

The button control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_SETWINDOWPARAMS

---

## Chapter 12. Entry Field Control Window Processing

This system-provided window procedure processes the actions on an entry field control (WC\_ENTRYFIELD).

### Purpose

An entry field control is a rectangular window that displays a single line of text that the operator can edit. When it has the focus, the cursor marks the current **insertion** or **replacement** point.

When working with entry fields, the WM\_CONTROL message is of major concern. An entry-field control communicates with its owner by sending WM\_CONTROL messages. It contains a notification code in MP1 and a handle to the current entry field in MP2. The return value for WM\_CONTROL is 0. Notification codes are denoted by an EN prefix.

---

### Entry Field Control Styles

These entry field control styles are available:

- |                      |   |
|----------------------|---|
| <b>ES_LEFT</b>       | The text in the control is left-justified. This is the default style if neither ES_RIGHT nor ES_CENTER is specified.  |
| <b>ES_RIGHT</b>      | The text in the control is right-justified.   |
| <b>ES_CENTER</b>     | The text in the control is centered.  |
| <b>ES_AUTOSIZE</b>   | The text will be sized to make sure the contents fit.   |
| <b>ES_AUTOSCROLL</b> | If the user tries to move off the end of a line, the control automatically scrolls one-third the width of the window in the appropriate direction.  |
| <b>ES_MARGIN</b>     | <p>This style can be used to cause a border to be drawn around the control, with a margin around the editable text. The margin is half a character-width wide and half a character-height high.</p> <p>When an entry field control with this style is positioned, it adjusts the position so that the text is placed at the position specified. This position differs from the original position by the width of the border and the margin.</p> |
| <b>ES_READONLY</b>   | <p>This style causes a single line entry field to be created in read only state.</p> <p>When an entry field is in read only state, characters do not get inserted into the text. However the insertion interface is still functional.</p> <p>The entry field read only state can be altered by use of the EM_SETREADONLY message.</p>   |

- ES\_UNREADABLE** This style causes the text to be displayed as an asterisk for each character. It can be used for passwords.
- ES\_COMMAND** This style identifies the entry field as a command entry field. This information is used by the Help Manager to provide command help if the end user requests help for this field.
- Not more than one entry field on each dialog should be given this style.
- ES\_AUTOTAB** This style indicates that when the field is filled by adding a character to the end of the entry field text, the effect of a tab key will be generated. Inserting or replacing a character in the middle of the text, however, does not result in an autotab.
- This style is recommended for use with fixed-length, non-scrollable fields that are filled completely. The maximum length of the entry field text is held in the control data, see "Entry Field Control Data" on page 12-3

These entry field controls are intended for countries that use a double-byte character encoding scheme:

- ES\_SBCS** The text is purely single-byte.
- If the number of characters entered exceeds EM\_SETTEXTLIMIT, or a DBCS character is entered, the alarm sounds and the last character entered is ignored.
- ES\_DBCS** The text is purely double byte.
- If the number of bytes in the entry field exceeds EM\_SETTEXTLIMIT, or an SBCS character is entered, the alarm sounds and the last character entered is ignored.
- ES\_ANY** The text is a mixture of SBCS and DBCS characters.
- If the number of bytes in the input field exceeds EM\_SETTEXTLIMIT, the alarm sounds and the last character entered is ignored.
- ES\_ANY is the default.
- Note:** If the queue code page is an ASCII code page and the data in the entry field is to be converted to an EBCDIC code page, there is a possibility that shift-in and shift-out characters introduced by the conversion process can cause the converted data to overrun the target field. Coding ES\_MIXED protects the target field from overrun in this situation.
- ES\_MIXED** The text is a mixture of SBCS and DBCS characters which may subsequently be converted from an ASCII DBCS code page to an EBCDIC DBCS code page with a consequent possible increase in the length of the data.

If

$DBCSChars * 2 + SBCSChars + N > EM\_SETTEXTLIMIT$

where N starts at 0 and is incremented whenever the string goes from SBCS to DBCS or DBCS to SBCS, the alarm sounds and the last character entered is ignored.

**Note:** For every conversion from SBCS to DBCS there must be a corresponding return to SBCS (N must be an even number).

---

## Entry Field Control Data

See "ENTRYFDATA" on page A-64.

---

## Default Colors

The following system colors are used when the system draws button controls:

SYSCLR\_ENTRYFIELD  
SYSCLR\_BUTTONDARK  
SYSCLR\_BUTTONLIGHT  
SYSCLR\_OUTPUTTEXT  
SYSCLR\_WINDOWTEXT  
SYSCLR\_HIGHLIGHTFOREGROUND  
SYSCLR\_HIGHLIGHTBACKGROUND

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_FOREGROUNDCOLOR  
PP\_DISABLEDFOREGROUND  
PP\_HIGHLIGHTFOREGROUND  
PP\_FONTNAMESIZE

---

## Entry Field Control Notification Messages

This message is initiated by the entry field control window to notify its owner of significant events.

---

### WM\_CONTROL (in Entry Fields)

For the cause of this message, see "WM\_CONTROL" on page 10-39.

#### Parameters

##### param1

**id** (USHORT)

Control window identity.

**usnotifycode** (USHORT)

Notify code.

- |                     |   |
|---------------------|---|
| <b>EN_CHANGE</b>    | The content of the entry field control has changed, and the change has been displayed on the screen.  |
| <b>EN_KILLFOCUS</b> | The entry field control is losing the focus.  |
| <b>EN_MEMERROR</b>  | The entry field control cannot allocate the storage necessary to accommodate window text of the length implied by the <b>EM_SETTEXTLIMIT</b> message.   |
| <b>EN_OVERFLOW</b>  | The entry field control cannot insert more text than the current text limit. The text limit may be changed with the <b>EM_SETTEXTLIMIT</b> message.<br><br>If the recipient of this message returns TRUE, then the entry field control retries the operation, otherwise it terminates the operation.  |
| <b>EN_SCROLL</b>    | The entry field control is about to scroll horizontally. This can happen in these circumstances: <ul style="list-style-type: none"><li>• The application has issued a WinScrollWindow call</li><li>• The content of the entry field control has changed</li><li>• The caret has moved</li><li>• The entry field control must scroll to show the caret position.</li></ul> |
| <b>EN_SETFOCUS</b>  | The entry field control is receiving the focus.   |

##### param2

**hwndcontrolspect** (HWND)

Entry field control window handle.

**Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

**Remarks**

The entry field control window procedure generates this message and sends it to its owner, informing the owner of the event.

**Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

**Related Messages**

- WM\_CONTROL



---

## Entry Field Control Window Messages

This section describes the entry field control window procedure actions on receiving these messages:

---

### EM\_CLEAR

This message deletes the text that forms the current selection.

#### Parameters

**param1**

**ulReserve** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserve** (ULONG)

Reserved value, should be 0.

#### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

#### Remarks

The entry field control window procedure responds to this message by deleting the text that forms the current selection and setting *usmaxsel* equal to *usminsel*.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

### EM\_COPY

This message copies the current selection to the clipboard.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

## **param2**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **rc (BOOL)**

Success indicator.

TRUE    Successful completion  
FALSE   Error occurred.

## **Remarks**

The entry field control window procedure responds to this message by copying the text that forms the current selection to the clipboard in CF\_TEXT format.

## **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## **EM\_CUT**

This message copies the text that forms the current selection to the clipboard, and then deletes it from the entry field control.

## **Parameters**

### **param1**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

### **param2**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **rc (BOOL)**

Success indicator.

TRUE    Successful completion  
FALSE   Error occurred.

## Remarks

The entry field control window procedure responds to this message by copying the text that forms the current selection to the clipboard in CF\_TEXT format, and then deleting it from the entry field control and setting *usmaxsel* equal to *usminsel*.

This message is the combination of a EM\_COPY message followed by a EM\_CLEAR message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_PASTE

This message replaces the text that forms the current selection with text from the clipboard.

## Parameters

### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### rc (BOOL)

Success indicator.

TRUE Successful completion  
FALSE Error occurred.

For example, if the text to be inserted does not fit in the entry field control without overflowing the text limit set by the EM\_SETTEXTLIMIT message, in which instance no text is inserted.

## Remarks

The entry field control window procedure responds to this message by replacing the text that forms the current selection with text from the clipboard, if the data is in CF\_TEXT format.

Only characters from the clipboard up to the first carriage return are used in the replacement.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_QUERYCHANGED

This message enquires if the text of the entry field control has been changed since the last enquiry.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Changed indicator.

**TRUE** The text in the entry field control has been changed since the last time it received this message or a WM\_QUERYWINDOWPARAMS message.

**FALSE** All other situations.

### Remarks

The entry field control window procedure responds to this message by setting *rc* to indicate whether the text of the entry field has been changed since the last time either this message or a WM\_QUERYWINDOWPARAMS (in Entry Fields) message has been received.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_QUERYFIRSTCHAR

This message returns the zero-based offset of the first character visible at the left edge of an entry-field control.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

## param2

### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### sOffset (SHORT)

Zero-based offset.

## Remarks

The entry field control window procedure responds to this message by returning the zero-based offset into the text that corresponds to the first character displayed in the entry field control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sOffset* to the default value of 0.

---

## EM\_QUERYREADONLY

This message returns the read only state of an entry field control.

## Parameters

### param1

#### ulReserved (ULONG)

Reserved value, should be 0.

### param2

#### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### rc (BOOL)

Read only state indicator.

TRUE Read only state is enabled.

FALSE Read only state is disabled.

## Remarks

The entry field control window procedure responds to this message by returning the read only state of the entry field control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_QUERYSEL

This message gets the zero-based offsets of the bounds of the text that forms the current selection.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### ReturnCode

**sMinSel** (SHORT)

Offset of the first character in the selection.

**sMaxSel** (SHORT)

Offset of the first character after the selection.

### Remarks

The entry field control window procedure responds to this message by returning the zero-based offsets of the bounds of the text that forms the current selection.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sMinSel* to the default value of 0, which is equivalent to setting both *sMinSel* and *sMaxSel* to 0.

---

## EM\_SETFIRSTCHAR

This message specifies the offset of the character to be displayed in the first position of the entry field control.

### Parameters

**param1**

**sOffset** (SHORT)

Zero-based offset of the first character to be displayed.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred. For example, because *sOffset* is not valid.

### Remarks

The entry field control window procedure responds to this message by setting the text displayed in the edit control so that the first character displayed on the left of the window has the zero-based index specified by *sOffset*.

An EN\_SCROLL notification message occurs, if the entry field control scrolls. This message returns FALSE if the edit control does not have the ES\_AUTOSCROLL style or it is center of right justified.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_SETINSERTMODE

This message sets the insert mode of an entry field.

### Parameters

#### param1

##### usInsert (USHORT)

Insert mode indicator.

TRUE    Enable insert mode.  
FALSE   Enable overtype mode.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Previous insert mode indicator.

TRUE    Insert mode was previously enabled.  
FALSE   Overtyping mode was previously enabled.

### Remarks

The entry field control window procedure responds to this message by setting the insert mode of the entry field, updating the SV\_INSERTMODE system constant and redrawing the entry field.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_SETREADONLY

This message sets the read only state of an entry field control.

### Parameters

#### param1

##### usReadOnly (USHORT)

Read only state indicator.

TRUE    Enable read only state  
FALSE   Disable read only state.



**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Previous read only state indicator.

TRUE Read only state was previously enabled.

FALSE Read only state was previously disabled.

## Remarks

The entry field control window procedure responds to this message by setting the read only state of the entry field control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_SETSEL

This message sets the zero-based offsets of the bounds of the text that forms the current selection.

## Parameters

**param1**

**usminsel** (USHORT)

Offset of the first character in the selection.

**usmaxsel** (USHORT)

Offset of the first character after the selection.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The entry field control window procedure responds to this message by setting the zero-based offsets of the bounds of the text that forms the current selection.

If *usminsel* equals *usmaxsel*, the current selection becomes an insertion point.

If *usminsel* equals 0 and *usmaxsel* is equal to or greater than the text limit set by the EM\_SETTEXTLIMIT message, the entire text is selected. Selected text is displayed in reverse color.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## EM\_SETTEXTLIMIT

This message sets the maximum number of bytes that an entry field control can contain.

## Parameters

**param1**

**sTextLimit** (SHORT)

Maximum number of characters in the entry field control.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred. For example, because not enough storage can be allocated.

## Remarks

The entry field control window procedure responds to this message by setting the maximum number of characters that can be contained.

This message is intended only to limit the length of lines that result from the user interacting with the entry field control. It also limits the length of text that can result from sending a EM\_PASTE or WM\_SETWINDOWPARAMS message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## WM\_CHAR (in Entry Fields)

For the cause of this message, see "WM\_CHAR" on page 10-32.

For a description of the parameters, see "WM\_CHAR" on page 10-32.

### Remarks

The entry field control window procedure responds to this message by sending it to its owner if it has not processed the keystroke. This is the most common means by which the input focus is switched around the various controls in a dialog box.

Unlike other controls, the *usvk* field of the message "WM\_CHAR" on page 10-32 takes precedence over other fields only when the Shift key is pressed.

If this message contains a valid *usch* field of the message "WM\_CHAR" on page 10-32. that character is entered into the text in insert or overwrite mode.

The keystrokes processed by an entry field control are:

|                          |  |
|--------------------------|--|
| <b>Left arrow</b>        | Move the cursor one character to the left.   |
| <b>Right arrow</b>       | Move the cursor one character to the right.  |
| <b>Shift+Left arrow</b>  | Extend the selection by one character to the left.   |
| <b>Shift+Right arrow</b> | Extend the selection by one character to the right.  |
| <b>Home</b>              | Move the cursor to the beginning of the text.  |
| <b>End</b>               | Move the cursor to the end of the text.  |
| <b>Backspace</b>         | Delete the character to the left of the cursor.  |
| <b>Delete</b>            | When the selection is an insertion point, delete the character to the right of the cursor, otherwise delete the current selection, but do not put it in the clipboard. |
| <b>Shift+Del</b>         | Cut the current selection to the clipboard.  |
| <b>Shift+Ins</b>         | Replace the current selection with the text contents from the clipboard.   |
| <b>Ctrl+Del</b>          | Delete to the end of the field.  |
| <b>Ctrl+Ins</b>          | Copy the current selection to the clipboard.   |

If the control contains more text than can be shown, the actions defined above that move the cursor cause the text to be scrolled. The amount of scrolling varies from key to key, and the position of the text within the control varies for the same cursor position.

## Default Processing

The default window procedure sends the message to the owner window if it exists, otherwise it takes no action on this message other than to set *rc* to FALSE.

## Related Messages

- WM\_CHAR

---

## WM\_QUERYCONVERTPOS (in Entry Fields)

For the cause of this message, see “WM\_QUERYCONVERTPOS” on page 10-72.

For a description of the parameters, see “WM\_QUERYCONVERTPOS” on page 10-72.

### Remarks

The entry field control window procedure updates *pCursorPos* to the position of the cursor and returns QCP\_CONVERT.

### Default Processing

For the default window procedure processing of this message see “WM\_QUERYCONVERTPOS” on page 10-72.

## Related Messages

- WM\_QUERYCONVERTPOS

---

## WM\_QUERYWINDOWPARAMS (in Entry Fields)

This message occurs when an application queries the entry field control window parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### Remarks

The entry field control window procedure responds to this message by returning the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure identified by the *pwndparams* parameter.

### Default Processing

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to 0 and sets *rc* to FALSE.

## Related Messages

- WM\_QUERYWINDOWPARAMS

---

## **WM\_SETWINDOWPARAMS (in Entry Fields)**

This message occurs when an application sets or changes the entry field control window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### **Remarks**

The entry field control window procedure responds to this message by setting the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure, identified by the *pwndparams* parameter.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_SETWINDOWPARAMS

---

## Chapter 13. Frame Control Window Processing

This system-provided window procedure processes the actions on a frame window (WC\_FRAME). The frame control window procedure sends all messages not processed to FID\_CLIENT and sets **reply** to 0.

### Purpose

The window that contains all of the parts listed below is called the *frame window*. Each of the parts that make up a window, such as the title bar and menu, are separate child windows of the frame window. All of these child windows, except the client window (FID\_CLIENT), are called *frame controls*.

FID\_CLIENT is not a frame control, it is an instance of a window class implemented by the application.

The frame window and all of the frame controls are implemented with system-provided preregistered window classes.

The frame window holds together all of the frame controls and FID\_CLIENT that make up an application window. The frame window is responsible for arranging the frame controls and the FID\_CLIENT as the frame window is sized and moved. It is also responsible for routing specific messages to its frame controls and the FID\_CLIENT.

Each of the frame controls and FID\_CLIENT are known to the frame window by a system-provided window-identifier value as listed below:

|                       |                       |
|-----------------------|-----------------------|
| <b>FID_CLIENT</b>     | Client window         |
| <b>FID_HORZSCROLL</b> | Horizontal scroll bar |
| <b>FID_MENU</b>       | Application menu      |
| <b>FID_MINMAX</b>     | Minimize/Maximize box |
| <b>FID_SYSMENU</b>    | System menu           |
| <b>FID_TITLEBAR</b>   | Title bar             |
| <b>FID_VERTSCROLL</b> | Vertical scroll bar.  |

For correct operation, only one window per frame must be defined with each of the above FID\_\* values.

---

### Frame Creation Flags

These frame creation flags are available:

|                      |                                |
|----------------------|--------------------------------|
| <b>FCF_TITLEBAR</b>  | Title bar.                     |
| <b>FCF_SYSMENU</b>   | System menu.                   |
| <b>FCF_MENU</b>      | Application menu.              |
| <b>FCF_MINMAX</b>    | Minimize and Maximize buttons. |
| <b>FCF_MINBUTTON</b> | Minimize button.               |

|                            |   |
|----------------------------|---|
| <b>FCF_MAXBUTTON</b>       | Maximize button.  |
| <b>FCF_VERTSCROLL</b>      | Vertical scroll bar.  |
| <b>FCF_HORZSCROLL</b>      | Horizontal scroll bar.  |
| <b>FCF_SIZEBORDER</b>      | Sizing border.  |
| <b>FCF_BORDER</b>          | Window is drawn with a thin border.   |
| <b>FCF_DLGBORDER</b>       | Window is drawn with a standard dialog border.  |
| <b>FCF_ACCELTABLE</b>      | Causes an accelerator table to be loaded, for this frame window, from the resource file identified on the WinCreateStdWindow function.  |
| <b>FCF_ICON</b>            | <p>Window is created with an icon associated with it that is used to represent the window when it is minimized.</p> <p>If present, the <i>Resource</i> parameter of the WinCreateStdWindow function must be the identity of an icon. This icon is loaded and associated with the window. When the window is minimized, the icon is shown if the screen is capable of showing it. When the window is destroyed, the icon is also destroyed.</p>  |
| <b>FCF_SHELLPOSITION</b>   | The window is created with a size and position determined by the shell, rather than explicitly by the application.  |
| <b>FCF_SYSMODAL</b>        | The frame window is System Modal.   |
| <b>FCF_NOBYTEALIGN</b>     | <p>When this flag is <b>not</b> set, the frame window is adjusted so that window operations, such as moving, can be performed in an optimized manner. For example, some displays can move a window more quickly if the movement is by a multiple of eight pels.</p> <p>If this flag is set, such optimizations are not performed and size and position values are honored.</p>  |
| <b>FCF_TASKLIST</b>        | <p>When this flag is set, the program title is added to the front of the frame window text, the resulting string is used as the window title and is also entered on the task list.</p> <p>In this context, the program title is the text string used by the Desktop Manager to identify the program, or the text string specified as a parameter in the START command. If neither string has been defined, the filename and extension of the .EXE file are used as the program title.</p> <p>Note that a WinSetWindowText will not change the entry in the switch list, a WinChangeSwitchEntry must be done to affect this.</p> |
| <b>FCF_NOMOVEWITHOWNER</b> | The window should not be moved when its owner is moved.   |

|                        |  |
|------------------------|--|
| <b>FCF_STANDARD</b>    | Same as (FCF_TITLEBAR   FCF_SYSMENU   FCF_MINBUTTON   FCF_MAXBUTTON   FCF_SIZEBORDER   FCF_ICON   FCF_MENU   FCF_ACCELTABLE   FCF_SHELLPOSITION   FCF_TASKLIST).<br><br>This value is assumed if any Frame Window is created with no Control Data. |
| <b>FCF_SCREENALIGN</b> | See FS_SCREENALIGN.  |
| <b>FCF_MOUSEALIGN</b>  | See FS_MOUSEALIGN.   |
| <b>FCF_AUTOICON</b>    | Performance optimization. When repainting iconized frames, the system will redraw the icon and will not send a WM_PAINT message to the application.  |
| <b>FCF_HIDEBUTTON</b>  | Hide button.   |
| <b>FCF_HIDEMAX</b>     | Hide and maximize buttons.   |

---

## Frame Control Styles

These frame control styles are available. Frame styles may only be used when the frame is created from a dialog template.

|                           |   |
|---------------------------|---|
| <b>FS_SCREENALIGN</b>     | The coordinates specifying the location of the dialog box are relative to the top left corner of the screen, rather than being relative to the owner window's origin.   |
| <b>FS_MOUSEALIGN</b>      | The coordinates specifying the location of the dialog box are relative to the position of the pointing device pointer at the time the window was created. The operating system tries to keep the dialog box on the screen, if possible. |
| <b>FS_SIZEBORDER</b>      | See FCF_SIZEBORDER.   |
| <b>FS_BORDER</b>          | See FCF_BORDER.   |
| <b>FS_DLGBORDER</b>       | See FCF_DLGBORDER.  |
| <b>FS_SYSMODAL</b>        | See FCF_SYSMODAL.   |
| <b>FS_NOBYTEALIGN</b>     | See FCF_NOBYTEALIGN.  |
| <b>FS_TASKLIST</b>        | See FCF_TASKLIST.   |
| <b>FS_NOMOVEWITHOWNER</b> | See FCF_NOMOVEWITHOWNER.  |
| <b>FS_AUTOICON</b>        | See FCF_AUTOICON.   |

---

## Frame Control Data

See "FRAMECDATA" on page A-99.



---

## Default Colors

The following system colors are used when the system draws button controls:

- SYSCLR\_DIALOGBACKGROUND
- SYSCLR\_ACTIVETITLE
- SYSCLR\_INACTIVETITLE
- SYSCLR\_APPWORKSPACE
- SYSCLR\_ACTIVEBORDER
- SYSCLR\_WINDOW
- SYSCLR\_SHADOW
- SYSCLR\_WINDOWFRAME
- SYSCLR\_FIRST.

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_BACKGROUND\_COLOR  
PP\_SHADOW  
PP\_FOREGROUND\_COLOR  
PP\_BORDER\_COLOR  
PP\_DISABLED\_BACKGROUND\_COLOR.

---

## Frame Control Notification Messages

These messages are initiated by the frame control window to notify the FID\_CLIENT window.

---

### WM\_MINMAXFRAME (in Frame Controls)

For the cause of this message, see “WM\_MINMAXFRAME” on page 10-58.

For a description of the parameters, see “WM\_MINMAXFRAME” on page 10-58.

#### Remarks

The window words QWS\_XRESTORE, QWS\_YRESTORE, QWS\_CXRESTORE, and QWS\_CYRESTORE for *hwnd* are initialized before this message is sent. The window state has not been changed when this message is sent, and so the WinQueryWindowPos function can be used.

This message is sent by default to the FID\_CLIENT window.

The system default actions, if FALSE is returned to this message, are based on the operation specified by the *pswp* parameter.

These actions affect the status of the frame window, and the title button windows and system menu windows contained within it, as follows:

- Window is maximized from a minimized state.
  - Title button windows:

The RESTORE button window is replaced by a MIN button window and the MAX button window is replaced by a RESTORE button window.
  - System menu window:

The MINIMIZE menu entry is enabled and the MAXIMIZE menu entry is disabled.
  - Other changes:

The frame window has the WS\_MAXIMIZED style bit set and the WS\_MINIMIZED style bit reset. Also the MS\_VERTICALFLIP style bit of the system menu window is reset.
- Window is restored from a minimized state.
  - Title button windows:

The RESTORE button window is replaced by a MIN button window (the MAX button window is unaltered).
  - System menu window:

The MINIMIZE menu entry is enabled, the RESTORE menu entry is disabled and the SIZE menu entry is enabled.

- Other changes:
 

The frame window has the WS\_MINIMIZED style bit and the MS\_VERTICALFLIP style bit of the system menu window reset.
- Window is minimized from a maximized state.
  - Title button windows:
 

The RESTORE button window is replaced by a MAX button window and the MIN button window is replaced by a RESTORE button window.
  - System menu window:
 

The MAXIMIZE menu entry is enabled and the MINIMIZE menu entry is disabled.
  - Other changes:
 

The frame window has the WS\_MINIMIZED style bit set and the WS\_MAXIMIZED style bit reset. Also the MS\_VERTICALFLIP style bit of the system menu window is set.
- Window is restored from a maximized state.
  - Title button windows:
 

The RESTORE button window is replaced by a MAX button window (the MIN button window is unaltered).
  - System menu window:
 

The MAXIMIZE menu entry is enabled, the RESTORE menu entry is disabled and the SIZE menu entry is enabled.
  - Other changes:
 

The frame window has the WS\_MAXIMIZED style bit reset.
- Window is minimized from a restored state.
  - Title-button windows:
 

The MIN button window is replaced by a RESTORE button window (the MAX button window is unaltered).
  - System menu window:
 

The RESTORE menu entry is enabled, the MINIMIZE menu entry is disabled and the SIZE menu entry is disabled.
  - Other changes:
 

The frame window has the WS\_MINIMIZED style bit set, and the MS\_VERTICALFLIP style bit of the system menu window is set.

- Window is maximized from a restored state.
  - Title-button windows:

The MAX button window is replaced with a RESTORE button window (the MIN button window is unaltered).
  - System menu window:

The RESTORE menu entry is enabled, the MAXIMIZE menu entry is disabled.
  - Other changes:

The frame window has the WS\_MAXIMIZED style bit set.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_MINMAXFRAME

---

## Frame Control Window Messages

This section describes the frame control window procedure actions on receiving the following messages.

---

### WM\_ACTIVATE (in Frame Controls)

For the cause of this message, see "WM\_ACTIVATE" on page 10-5.

For a description of the parameters, see "WM\_ACTIVATE" on page 10-5.

#### Remarks

The frame control window procedure responds to this message by first sending a TBM\_SETHILITE message to the FID\_TITLEBAR control, if it exists, to highlight or unhighlight the title bar. If the style is FCF\_DLGBORDER, the border is redrawn in either highlighted or unhighlighted state, as necessary.

It then sends the WM\_ACTIVATE message to the FID\_CLIENT window.

Then it sets *ulReserved* to 0.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_ACTIVATE

---

### WM\_ADJUSTFRAMEPOS

This message is sent to a frame window whose position or size is to be adjusted.

#### Parameters

**param1**

**pswp** (PSWP)

New frame window state.

This points to a SWP structure.

The structure has been filled in by the WinSetWindowPos or WinSetMultWindowPos functions with the proposed move or size data for the frame window.

**param2**

**hsavewphswwp** (HSAVEWP)

Identifier of the frame window repositioning process.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

When a `WinSetWindowPos` or `WinSetMultWindowPos` function involves adjusting the position or size of a frame window, a `WM_ADJUSTFRAMEPOS` message is sent to the frame window.

The frame control processes the message by informing all the windows in its owner hierarchy, that is all the windows owned by the frame and all the windows owned by them and so on, by sending each a `WM_OWNERPOSCHANGE` message. Each window receiving the a `WM_OWNERPOSCHANGE` message is expected to modify the `SWP` structure provided as the first parameter in the message to the appropriate values relative to the new position and/or size of its owner, whose new position and size is specified in a `SWP` structure provided as the second parameter in the message.

In this way the frame control can determine the state changes to be made to all the windows in its owner hierarchy, in accordance with the values specified in the `SWP` structure referenced by the *pswp* parameter. The rules for changing the state of these owned windows are:

`SWP_SIZE` and `SWP_MOVE`

The owned window is moved relative to the top left corner of its owner.

`SWP_SHOW`

The visibility state of an owned window is changed to agree with that of their owner.

`SWP_MINIMIZE`

An owned window is made invisible when the owner is minimized.

`SWP_MAXIMIZE` and `SWP_RESTORE`

An owned window that was previously made invisible when the owner was minimized is made visible.

The frame window coordinates the repositioning of the frame window and all its owned windows, by using the `WinSaveWindowPos` function to associate those windows whose states are to change with the identifier of the frame window repositioning process, that is the *hsavewphsvwp* parameter. Eventually, the state changes to be made to the owned windows are contained in the array of `SWP` structures identified by the *pswp* parameter.

If the frame window is subclassed, this message must then be passed to the superclass window procedure for processing. The superclass window procedure is the window procedure of the window before it was subclassed. This message is passed along the chain of window procedures and is eventually processed by the system frame window procedure.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## **WM\_BUTTON1DBLCLK (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON1DBLCLK” on page 10-12.

For a description of the parameters, see “WM\_BUTTON1DBLCLK” on page 10-12.

### **Default Processing**

If the frame is minimized, the frame control window procedure causes the frame window to return to its previous state. Otherwise, the message is handled like a WM\_BUTTON1DOWN message.

### **Related Messages**

- WM\_BUTTON1DBLCLK

---

## **WM\_BUTTON2DBLCLK (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON2DBLCLK” on page 10-18.

For a description of the parameters, see “WM\_BUTTON2DBLCLK” on page 10-18.

### **Default Processing**

The frame control window procedure processes this message identically to WM\_BUTTON1DBLCLK (in Frame Controls).

### **Related Messages**

- WM\_BUTTON2DBLCLK

---

## **WM\_BUTTON1DOWN (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON1DOWN” on page 10-13.

For a description of the parameters, see “WM\_BUTTON1DOWN” on page 10-13.

### **Remarks**

This message is posted to the application queue associated with the window that is to receive the pointer button information.

### **Default Processing**

The frame control window procedure responds to this message by issuing the WinSetActiveWindow function and sets *rc* to TRUE. If this is over a part of the window that does not have a frame control, it issues a WinSetActiveWindow function. If the click is over the size border, this window begins tracking by sending a WM\_TRACKFRAME message to itself. If the click is not over the size border, this message is passed on.

### **Related Messages**

- WM\_BUTTON1DOWN

---

## **WM\_BUTTON2DOWN (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON2DOWN” on page 10-19.

For a description of the parameters, see “WM\_BUTTON2DOWN” on page 10-19.

### **Remarks**

This message is posted to the application queue associated with the window that is to receive the pointer button information.

### **Default Processing**

The frame control window procedure processes this message identically to “WM\_BUTTON1DOWN (in Frame Controls)” on page 13-10.

### **Related Messages**

- WM\_BUTTON2DOWN

---

## **WM\_BUTTON1UP (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON1UP” on page 10-16.

For a description of the parameters, see “WM\_BUTTON1UP” on page 10-16.

### **Remarks**

This message is posted to the application queue associated with the window that is to receive the pointer button information.

### **Default Processing**

The frame control window procedure responds to this message by issuing the WinSetActiveWindow function and sets *rc* to TRUE. If the window is not minimized, this message is not processed. If the frame is minimized, this message causes the system menu to pop up.

### **Related Messages**

- WM\_BUTTON1UP

---

## **WM\_BUTTON2UP (in Frame Controls)**

For the cause of this message, see “WM\_BUTTON2UP” on page 10-22.

For a description of the parameters, see “WM\_BUTTON2UP” on page 10-22.

### **Remarks**

This message is posted to the application queue associated with the window that is to receive the pointer button information.



## Default Processing

The frame control window procedure processes this message identically to “WM\_BUTTON1UP (in Frame Controls)” on page 13-11.

## Related Messages

- WM\_BUTTON2UP

---

## WM\_CALCFRAMERECT (in Frame Controls)

For the cause of this message, see “WM\_CALCFRAMERECT” on page 10-29.

For a description of the parameters, see “WM\_CALCFRAMERECT” on page 10-29.

## Remarks

Frame control calculates the appropriate rectangle, taking into account byte alignment, or nonbyte alignment if FCF\_NOBYTEALIGN is specified.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_CALCFRAMERECT

---

## WM\_CHAR (in Frame Controls)

This message is sent by controls to their owner window if they do not process the key stroke themselves. It is the most common means by which the input focus is switched around the various controls in a dialog box.

For a description of the parameters, see “WM\_CHAR” on page 10-32.

## Default Processing

The frame control window procedure responds to this message as follows:

- If the message contains a valid VK\_ value, that value is processed before any valid character in the message.
- If the character matches a mnemonic in the text of a button or static control child window, the focus is set to that window.
- If the character is Tab or Backtab, the focus is set to the next or previous tabstop window.
- If the character is Up or Left Arrow, the focus is set to the previous item in the group.
- If the character is Down or Right Arrow, the focus is set to the next item in the group.

- If the Enter key is pressed, a WM\_COMMAND message is posted to itself, containing the identity of the button with the focus, or, if none, the identity of the default push button.
- If the Escape key is pressed, a WM\_COMMAND message is posted to itself with the command value DID\_CANCEL.

### Related Messages

- WM\_CHAR

---

## WM\_CLOSE (in Frame Controls)

For the cause of this message, see “WM\_CLOSE” on page 10-35.

For a description of the parameters, see “WM\_CLOSE” on page 10-35.

### Remarks

Frame control sends this message to the client window (FID\_CLIENT) if it exists, otherwise it calls the WinDefWindowProc function.

### Default Processing

The default window procedure posts a WM\_QUIT message to the appropriate queue and sets *ulReserved* to 0.

### Related Messages

- WM\_CLOSE

---

## WM\_COMMAND

For the cause of this message, see “WM\_COMMAND” on page 10-37.

For a description of the parameters, see “WM\_COMMAND” on page 10-37.

### Default Processing

The Frame Control window procedure responds to this message by sending it the client window if it exists, otherwise the message is thrown away.

---

## WM\_DRAWITEM (in Frame Controls)

For the cause of this message, see “WM\_DRAWITEM” on page 10-42.

For a description of the parameters, see “WM\_DRAWITEM” on page 10-42.

### Remarks

The identity of the top-level action-bar menu that generated this message is found. If the identity is FID\_MENU, the message is passed to the window with identity FID\_CLIENT.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_DRAWITEM

---

## WM\_ERASEBACKGROUND

This message causes a client window to be filled with the background, should this be appropriate.

## Parameters

### param1

#### **hpsFrame** (HPS)

Presentation-space handle for the frame window.

### param2

#### **pprcPaint** (PRECTL)

Rectangle structure of rectangle to be painted.

This points to a RECTL structure.

## Returns

### rc (BOOL)

Processed indicator.

- TRUE** If a FID\_CLIENT window exists, the area of the frame covered by the FID\_CLIENT window is erased in the system-window background color.
- If no FID\_CLIENT window exists, the entire frame window is erased in the system-window background color.
- FALSE** The client window did process the message.

## Remarks

The frame window procedure processes this message in the following manner:

1. The frame window sends this message to the client in response to the frame WM\_PAINT message, with the presentation-space handle of the frame window (obtained from WinBeginPaint).
2. If the client window returns TRUE, the frame window procedure erases the rectangle of the frame window covered by the client window, by filling it with the system color SCLR\_WINDOW.
3. If the client window returns FALSE, no action is taken. This is the default behavior, as WinDefWindowProc returns FALSE if passed this message.

- Also, the client window can use the presentation-space handle passed in this message to selectively erase parts of the screen. If the client window processes the message in this way, `FALSE` should be returned to avoid the erasure being done automatically by the frame window procedure.

It should be noted again that the presentation space is *not* a client window presentation space; it is a presentation space for the frame window returned by `WinBeginPaint`, that is, a cached presentation space in frame (not client) window coordinates, clipped to the area of the frame that needs to be updated (possibly including areas outside the client window).

## Default Processing

The default window procedure takes no action on this message, other than to set `rc` to `FALSE`.

---

## WM\_FLASHWINDOW

An application has issued a `WinFlashWindow` function.

### Parameters

**param1**

**usFlash** (USHORT)

Flash indicator.

TRUE Start the window border flashing

FALSE Stop the window border flashing.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Default Processing

The frame control window procedure responds to this message from an application by starting or stopping the flashing of the window border, and by setting `rc` as appropriate.

---

## WM\_FOCUSCHANGE (in Frame Controls)

For the cause of this message, see “WM\_FOCUSCHANGE” on page 10-47.

For a description of the parameters, see “WM\_FOCUSCHANGE” on page 10-47.

### Remarks

The frame control responds to this message by sending the other messages depending on the value of the *fsFocusChange* parameter. These messages, if sent, are sent in the following order:

1. WM\_SETFOCUS to the window losing the focus.
2. WM\_SETSELECTION to the windows losing their selection.
3. WM\_ACTIVATE to the windows being deactivated.
4. WM\_ACTIVATE to the windows being activated.
5. WM\_SETSELECTION to the windows being selected.
6. WM\_SETFOCUS to the window receiving the focus.

### Default Processing

The default window procedure sends this message to either the owner, if one exists, or to the parent of the window, if it is not the desktop window, otherwise it sets *ulReserved* to 0.

### Related Messages

- WM\_FOCUSCHANGE

---

## WM\_FORMATFRAME (in Frame Controls)

For the cause of this message, see “WM\_FORMATFRAME” on page 10-48.

For a description of the parameters, see “WM\_FORMATFRAME” on page 10-48.

### Remarks

Applications that subclass frame controls may find that the frame is already subclassed; the number of frame controls is variable.

The WM\_FORMATFRAME and WM\_QUERYFRAMECTLCOUNT messages must always be subclassed by calling the previous window procedure and modifying its result.

### Default Processing

The SWP structure for the FID\_CLIENT frame control, if present, is the last element of the *pswp* parameter, unless additional frame controls are added by subclassing; the SWP structures for these follow that for FID\_CLIENT if present. The frame control window procedure first sends the message to the FID\_CLIENT window. If FID\_CLIENT returns *ccount* to indicate that the message has been processed, no additional processing is performed.

If not processed by the client, the frame control window procedure calculates the size and position of all the standard frame controls.

## Related Messages

- WM\_FORMATFRAME

---

## WM\_INITMENU (in Frame Controls)

For the cause of this message, see “WM\_INITMENU” on page 10-53.

For a description of the parameters, see “WM\_INITMENU” on page 10-53.

### Remarks

The identity of the top-level action-bar menu that generated this message is found. If the identity is FID\_MENU, the message is passed to the window with identity FID\_CLIENT. If the identity is FID\_SYSMENU the system menu state is initialized according to the current state of the window.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_INITMENU

---

## WM\_MEASUREITEM (in Frame Controls)

For the cause of this message, see “WM\_MEASUREITEM” on page 10-55.

For a description of the parameters, see “WM\_MEASUREITEM” on page 10-55.

### Remarks

The identity of the top-level action bar menu that generated this message is found. If the identity is FID\_MENU, the message is passed to the window with identity FID\_CLIENT.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sHeight* to the default value of 0.

## Related Messages

- WM\_MEASUREITEM

---

## WM\_MENUSELECT (in Frame Controls)

For the cause of this message, see “WM\_MENUSELECT (in Frame Controls).”

For a description of the parameters, see “WM\_MENUSELECT (in Frame Controls).”

### Remarks

The identity of the top-level action-bar menu that generated this message is found. If the identity is FID\_MENU, the message is passed to the window with identity FID\_CLIENT.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to TRUE.

### Related Messages

- WM\_MENUSELECT

---

## WM\_NEXTMENU (in Frame Controls)

For the cause of this message, see “WM\_NEXTMENU” on page 10-63.

For a description of the parameters, see “WM\_NEXTMENU” on page 10-63.

### Remarks

The frame control window procedure processes the message by returning the handle of the system menu window if *hwndMenu* is the handle of the main action bar window, or by returning the handle of the main action bar window if *hwndMenu* is the handle of the system menu window.

### Default Processing

The default window procedure takes no action on this message, other than to set *hwndNewMenu* to NULLHANDLE.

### Related Messages

- WM\_NEXTMENU

---

## WM\_OWNERPOSCHANGE

This message is sent by a frame window processing the WM\_ADJUSTFRAMEPOS message.

## Parameters

### param1

#### **ppswp** (PSWP)

Owned window state.

This points to a SWP structure.

The receiver of this message is expected to alter this SWP parameter to the appropriate values relative to the new position and/or size of its owner, whose new position and size is specified in a SWP structure in the *ppswpOwner* parameter.

### param2

#### **ppswpOwner** (PSWP)

Owner window state.

This points to a SWP structure.

This represents the new position and size of the owner window.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_PAINT (in Frame Controls)

For the cause of this message, see "WM\_PAINT" on page 10-66.

For a description of the parameters, see "WM\_PAINT" on page 10-66.

## Default Processing

The frame is redrawn as governed by the FCF\_BORDER or FCF\_DLGBORDER style. A WM\_ERASEBACKGROUND message is sent to FID\_CLIENT window, and if it returns FALSE, then the FID\_CLIENT window is erased to the system-provided window background color and sets *ulReserved* to 0.

## Related Messages

- WM\_PAINT



---

## WM\_QUERYBORDERSIZE

This message is sent to the frame window to determine the width and height of the border of the window.

### Parameters

#### param1

##### pSize (PWPOINT)

Width and height of size border control.

This points to a POINTL structure, that is used to hold the width in the x parameter and the height in the y parameter.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Remarks

The frame window responds to this message by returning the width and height of its border in the *pSize* parameter, as follows:

- SV\_CX/CYSIZEBORDER if FCF\_SIZEBORDER is specified
- SV\_CX/CYDLGFRAME if FCF\_DLGBORDER is specified
- SV\_CX/CYBORDER if FS\_BORDER is specified.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## WM\_QUERYCONVERTPOS (in Frame Controls)

For the cause of this message, see "WM\_QUERYCONVERTPOS" on page 10-72.

For a description of the parameters, see "WM\_QUERYCONVERTPOS" on page 10-72.

### Remarks

The frame control window procedure returns QCP\_NOCONVERT.

## Default Processing

For the default window procedure processing of this message see WM\_QUERYCONVERTPOS

## Related Messages

- WM\_QUERYCONVERTPOS

---

## WM\_QUERYFOCUSCHAIN

This message is used to request the handle of a window in the focus chain.

### Parameters

**param1**

#### fsCmd (USHORT)

Command to be performed.

This field contains a flag to indicate what action is to be performed:

**QFC\_NEXTINCHAIN** Return the next window in the focus chain.

The *hwndParent* parameter is not used.

**QFC\_ACTIVE**

Return the handle of the frame window that would be activated or deactivated, if this window gains or loses the focus.

The window handle returned is a child of the window specified by the *hwndParent* parameter.

**QFC\_FRAME**

Return the handle of the first frame window associated with this window.

The *hwndParent* parameter is not used.

**QFC\_SELECTACTIVE**

Return the handle of the window from the group of owned windows to which this window belongs which either currently has the focus or, if no window has the focus, previously had the focus.

Return NULL, if no window in the owner group has had the focus.

The *hwndParent* parameter is not used.

**QFC\_PARTOFCHAIN**

Return TRUE if the handle of the window identified by the *hwndParent* parameter is in the focus chain, otherwise return FALSE.

Because this message is passed along the focus chain, this is equivalent to returning TRUE, if the handle of the window receiving this message is *hwndParent* or to returning FALSE, if it is not.

**param2**

**hwndParent** (HWND)  
Parent window.

### Returns

**hwndResult** (HWND)

Handle of the window requested.

0 No window handle exists for this case of the *fsCmd* parameter  
This value is also to be interpreted as FALSE for the case when the *fsCmd* is set to QFC\_PARTOFCHAIN.

Other Handle of the window requested.  
This value is also to be interpreted as TRUE for the cases when the *fsCmd* is set to QFC\_PARTOFCHAIN.

### Remarks

The frame control window procedure responds to this message by returning the appropriate window handle, as described under the *fsCmd* field.

### Default Processing

The default window procedure takes the same action as the frame control window procedure.

---

## WM\_QUERYFRAMECTLCOUNT

This message is sent to the frame window in response to the receipt of a WM\_SIZE or a WM\_UPDATEFRAME (in Frame Controls) message.

### Parameters

**param1**

**ulReserved** (ULONG)  
Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Returns

**sControlCount** (SHORT)

Count of frame controls.

## Remarks

By sending this message to itself, any procedures that subclass the frame window become aware that the number of frame controls is being calculated and include any special frame controls of the subclass in the count.

This count is used to allocate the appropriate number of SWP structures that are passed in the WM\_FORMATFRAME (in Frame Controls) message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sControlCount* to the default value of 0.

---

## WM\_QUERYFRAMEINFO

This message enables an application to query information about frame windows.

## Parameters

**param1**

**ulReserved (ULONG)**

Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

**flFlags (ULONG)**

Frame information flags.

FI\_FRAME

Identifies a frame window.

FI\_OWNERHIDE

The frame window is hidden when its owner is hidden.

FI\_NOMOVEWITHOWNER

The frame window does not move with its owner.

FI\_ACTIVATEOK

The frame window may be activated. This means, for example, that the frame window is not disabled.

## Remarks

This message can be used to query whether or not a particular window is a frame window.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_QUERYICON

This message is sent to a frame window to query its associated icon.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**hptricon** (HPOINTER)

Handle to the icon.

### Default Processing

The icon for the frame is returned.

---

## WM\_QUERYWINDOWPARAMS (in Frame Controls)

This message occurs when an application queries the frame control window parameters.

For a description of the parameters, see "WM\_QUERYWINDOWPARAMS" on page 10-75.

### Default Processing

The frame control window procedure queries the appropriate window parameters in accordance with *pwndparams* and sets *rc* to TRUE if the operation is successful, otherwise to FALSE.

The window text of a frame control is obtained by sending this message to its FID\_TITLEBAR.

### Related Messages

- WM\_QUERYWINDOWPARAMS

---

## WM\_SETBORDERSIZE

This message is sent to the frame window to change the width and height of the border.

### Parameters

#### param1

**uscx** (USHORT)  
Width of border.

#### param2

**uscy** (USHORT)  
Height of border.

### Returns

#### rc (BOOL)

Success indicator.

TRUE     Successful completion  
FALSE    Error occurred.

### Remarks

The frame control sets the width and height to *uscx* and *uscy* respectively.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## WM\_SETICON

This message is sent to a frame window to set its associated icon.

### Parameters

#### param1

**hptrIcon** (HPOINTER)  
New icon handle.

#### param2

**ulReserved** (ULONG)  
Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Default Processing

The icon for the frame is set.

---

## WM\_SETWINDOWPARAMS (in Frame Controls)

This message occurs when an application sets or changes the frame control window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

## Default Processing

The frame control window procedure sets the appropriate window parameters in accordance with *wndparams* and sets *rc* to TRUE if the operation is successful, otherwise to FALSE.

The window text of a frame control is set by sending this message to its FID\_TITLEBAR.

## Related Messages

- WM\_SETWINDOWPARAMS

---

## WM\_SIZE (in Frame Controls)

For the cause of this message, see “WM\_SIZE” on page 10-88.

For a description of the parameters, see “WM\_SIZE” on page 10-88.

## Default Processing

The frame control window procedure responds to this message by sending a WM\_FORMATFRAME (in Frame Controls) message to itself and by setting *ulReserved* to 0.

## Related Messages

- WM\_SIZE

---

## WM\_SYSCOMMAND

This message occurs when a control window has a significant event to notify to its owner, or when a key stroke has been translated by an accelerator table into a WM\_SYSCOMMAND.

### Parameters

#### param1

##### uscmd (USHORT)

Command value.

The frame control takes the action described on these *uscmd* values:

|               |  |
|---------------|--|
| SC_SIZE       | Sends a WM_TRACKFRAME (in Frame Controls) to the frame window.   |
| SC_MOVE       | Sends a WM_TRACKFRAME (in Frame Controls) to the frame window.   |
| SC_MINIMIZE   | If a control with the identifier FID_MINMAX is present, minimizes the frame window, or restores it to a remembered size and position.  |
| SC_MAXIMIZE   | If a control with the identifier FID_MINMAX is present, maximizes the frame window, or restores it to a remembered size and position.<br><br>When a window is moved or sized in the normal way at least one border should remain on the screen. When a window is maximized and the maximum size is as large as the screen, all borders should be positioned just outside the screen. |
| SC_RESTORE    | If a control with the identifier FID_MINMAX is present, restores a maximized frame window to its previous size and position.   |
| SC_NEXT       | Cycles the active window status to the next main window.   |
| SC_APPMENU    | Sends a MM_STARTMENUODE message to the control with the identifier FID_MENU.   |
| SC_SYSMENU    | Sends a MM_STARTMENUODE message to the control with the identifier FID_SYSMENU.  |
| SC_CLOSE      | If Close is not enabled in the system menu, this message is ignored. Otherwise the frame posts a WM_CLOSE message to the client if it exists or to itself, if not.   |
| SC_NEXTFRAME  | The next frame window that is a child of the desktop window is activated.  |
| SC_NEXTWINDOW | The next window with the same owner window is activated.   |



|                 |  |
|-----------------|--|
| SC_TASKMANAGER  | The Task List is activated.  |
| SC_HELPEXTENDED | The frame manager sends HM_EXT_HELP to the associated Help Manager Object Window. If there is no such associated window, the original message is sent to the client.   |
| SC_HELPKEYS     | The frame manager sends HM_KEYS_HELP to the associated Help Manager Object Window. If there is no such associated window, the original message is sent to the client.  |
| SC_HELPINDEX    | The frame manager sends HM_HELP_INDEX to the associated Help Manager Object Window. If there is no such associated window, the original message is sent to the client. |
| SC_HIDE         | Sets the visibility state of the frame window to off causing it to appear hidden or invisible.   |

## param2

### ussource (USHORT)

Source type.

Identifies the type of control:

|                    |  |
|--------------------|--|
| CMDSRC_PUSHBUTTON  | Posted by a push-button control: <i>uscmd</i> is the window identifier of the push button.           |
| CMDSRC_MENU        | Posted by a menu control: <i>uscmd</i> is the identifier of the menu item.                           |
| CMDSRC_ACCELERATOR | Posted as the result of an accelerator: <i>uscmd</i> is the accelerator command value.               |
| CMDSRC_OTHER       | Other source: <i>uscmd</i> gives further control-specific information defined for each control type. |

### fpointer (BOOL)

Pointing-device indicator.

|       |   |
|-------|---|
| TRUE  | The message is posted as a result of a pointing-device operation. |
| FALSE | The message is posted as a result of a keyboard operation.        |

### ulReserved (ULONG)

Reserved value, should be 0.

## Remarks

This message is posted to the window procedure of the owner of the frame control. *ulReserved* is set to 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## WM\_TRACKFRAME (in Frame Controls)

This message is sent to a frame window whenever it is to be moved or sized.

### Parameters

**param1**

**fsTrackFlags** (USHORT)

Tracking flags.

Contains a combination of one or more TF\_\* flags; for details, see the TRACKINFO data structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred, or the operation is terminated.

### Remarks

The frame control window procedure responds to this message by causing a tracking rectangle to be drawn to move or size the window. For information, see the WinTrackRect function.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to TRUE.

### Related Messages

- WM\_TRACKFRAME

---

## **WM\_TRANSLATEACCEL (in Frame Controls)**

For the cause of this message, see “WM\_TRANSLATEACCEL” on page 10-95.

For a description of the parameters, see “WM\_TRANSLATEACCEL” on page 10-95.

### **Remarks**

The frame control window procedure processes the message by checking whether the character is in the accelerator table, by using the WinTranslateAccel function.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_TRANSLATEACCEL

---

## **WM\_TRANSLATEMNEMONIC (in Frame Controls)**

For the cause of this message, see “WM\_TRANSLATEMNEMONIC” on page 10-96.

For a description of the parameters, see “WM\_TRANSLATEMNEMONIC” on page 10-96.

### **Remarks**

The frame control window procedure processes the message by sending it to the application menu window, that is, the window with the identity FID\_MENU.

### **Default Processing**

For the default window procedure processing of this message, see “WM\_TRANSLATEMNEMONIC” on page 10-96.

### **Related Messages**

- WM\_TRANSLATEMNEMONIC

---

## **WM\_UPDATEFRAME (in Frame Controls)**

For the cause of this message, see “WM\_UPDATEFRAME” on page 10-97.

For a description of the parameters, see “WM\_UPDATEFRAME” on page 10-97.

### **Remarks**

This message must be sent to the frame window whenever an application adds or removes one of the frame controls identified by the FCF\_\* flags. It must also be sent if the application adds or removes a submenu of the menu bar of the frame window.

The frame control window procedure first sends the message on to the FID\_CLIENT window. The FID\_CLIENT window might either reformat the frame window and set *rc* to TRUE, in

which case the frame control window procedure takes no further action, or it might set *rc* to FALSE, in which case the frame control window procedure performs the reformatting.

If *flCreateFlags* contains FCF\_SIZEBORDER, reformatting the frame window includes invalidating the area occupied by the size border.

The frame control window procedure sets *rc* to TRUE.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to TRUE.

### **Related Messages**

- WM\_UPDATEFRAME



---

## Chapter 14. List Box Control Window Processing

This system-provided window procedure processes the actions on a list box control (WC\_LISTBOX).

### Purpose

A list box control is a window containing a list of items. Each item in a list box contains a text string (0 or more characters) and a handle. The text string is displayed in the list box window. The handle can be used by the application to refer to other data associated with each item.

---

### List Box Control Styles

These list box control styles are available:

- LS\_HORZSCROLL** The list box control enables the operator to scroll the list box horizontally.
- LS\_MULTIPLESEL** The list box control enables the operator to select more than one item at any one time. Lists that do not have this style allow only a single selection at any one time. If this style is specified, **LS\_EXTENDESEL** should also be specified.
- LS\_EXTENDESEL** If this style is specified, the extended selection user interface is enabled.
- LS\_OWNERDRAW** The list box control has one or more items that can be drawn by the owner. Typically, these items are represented by bit maps rather than by text strings.
- LS\_NOADJUSTPOS** If this style is included, the list box control is drawn at the size specified. This can cause parts of an item to be shown.

---

### List Box Control Data

None.

---

### Default Colors

The following system colors are used when the system draws button controls:

- SYSCLR\_FIELDBACKGROUND
- SYSCLR\_BUTTONDARK
- SYSCLR\_WINDOW
- SYSCLR\_WINDOWTEXT
- SYSCLR\_ENTRYFIELD
- SYSCLR\_HILITEFOREGROUND
- SYSCLR\_HILITEBACKGROUND
- SYSCLR\_WINDOWFRAME

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_DISABLEDFOREGROUND  
PP\_FOREGROUND  
PP\_HILITEFOREGROUND  
PP\_BORDER

---

## List Box Control Notification Messages

These messages are initiated by the list box control window to notify its owner of significant events.

---

### WM\_CONTROL (in List Boxes)

For the cause of this message, see “WM\_CONTROL” on page 10-39.

#### Parameters

##### param1

###### id (USHORT)

Control-window identity.

###### usnotifycode (USHORT)

Notify code.

The list box control window procedure uses these notification codes:

|              |  |
|--------------|--|
| LN_ENTER     | Either the Enter or Return key has been pressed while the list box control has the focus, or the list box control has been double-clicked. |
| LN_KILLFOCUS | The list box control loses the focus.  |
| LN_SCROLL    | The list box control is about to scroll horizontally. This can happen when the application has issued a WinScrollWindow function.          |
| LN_SETFOCUS  | The list box control receives the focus.   |
| LN_SELECT    | An item is being selected (or deselected).   |

**Note:** To discover the index of the selected item, the application must use the LM\_QUERYSELECTION message.

##### param2

###### hwndcontrolspect (HWND)

List box control window handle.

#### Returns

##### ulReserved (ULONG)

Reserved value, should be 0.



## Remarks

The list box control window procedure generates this message and sends it to its owner, informing the owner of this event.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_CONTROL

---

## WM\_DRAWITEM (in List Boxes)

This notification is sent to the owner of a list box control each time an item is to be drawn.

### Parameters

**param1**

**idListBox** (USHORT)  
Window identifier.

The window identity of the list box control sending this notification message.

**param2**

**pOwnerItem** (OWNERITEM)  
Owner-item structure.

This points to an owner-item structure; see "OWNERITEM" on page A-136.

### Returns

**rc** (BOOL)  
Item-drawn indicator.

**TRUE** The owner draws the item, so the list box control does not draw it.  
**FALSE** If the item contains text and the owner does not draw the item, the owner returns this value, and the list box control draws the item.

## Remarks

The list box control window procedure only draws items that are represented by text strings and emphasizes selected items by inverting them.

If an application uses list box controls containing items that are not represented by text strings, or requires that the emphasized state of an item is to be drawn in a special manner, the list box control must specify the style **LS\_OWNERDRAW** and those items must be drawn by the owner.

The list box control window procedure generates this message and sends it to the owner of the list box control, informing the owner that an item is to be drawn, offering the owner the opportunity to draw that item, and indicating that either the item has been drawn, or that the list box control is to draw it.

The item text must not be changed during the processing of this message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

### Related Messages

- WM\_DRAWITEM

---

## WM\_MEASUREITEM (in List Boxes)

This notification is sent to the owner of a list box control to establish the height and width for an item in that control.

### Parameters

**param1**

**sListBox** (SHORT)  
List-box identifier.

**param2**

**sItemIndex** (SHORT)  
Item index.  
The zero-based index of the item which has changed.

### Returns

**ReturnCode**

**sHeight** (SHORT)  
Height of item.

**sWidth** (SHORT)  
Width of item.

This value is required only if the list box control is scrollable horizontally, that is, it has a style of LS\_HORZSCROLL.

### Remarks

This message is sent to the owner of a list box that has a style of LS\_OWNERDRAW, to offer the owner an opportunity to establish the height and width (for a horizontally scrollable

list box control) of an item that accommodates any special requirements for the drawing of items in that list box. It is sent when items in the list box are inserted or deleted, and also when presentation parameters for the list box change.

All items in a list box must have the same height, which must be greater than or equal to the height of the current font.

In particular, this notification is sent to the owner of a list box that has a style of `LS_OWNERDRAW`, to offer the owner an opportunity to establish the height and width (for a horizontally scrollable list box control) of an item that accommodates any special requirements for the drawing of items in that list box.

### **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sHeight* to the default value of 0.

### **Related Messages**

- `WM_MEASUREITEM`

---

## List Box Control Window Messages

This section describes the list box control window procedure actions on receiving the following messages.

---

### LM\_DELETEALL

This message is sent to a list box control to delete all the items in the list box.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

#### Remarks

The list box control window procedure responds to this message by deleting all the items in the list box and by setting *rc* to TRUE.

#### Default Processing

The default window procedure does not expect to receive this message and, therefore, takes no action on it, other than to set *rc* to the default value of FALSE.

---

### LM\_DELETEITEM

This message deletes an item from the list box control.

#### Parameters

**param1**

**sltemIndex** (SHORT)

Item index.

The zero-based index of the item to be deleted.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**sItemsLeft** (SHORT)

Number remaining.

The number of items in the list after the item is deleted.

### Remarks

The list box control window procedure responds to this message by deleting the indexed item of the list box and by setting *sItemsLeft* to the count of the items in the list after the item is deleted.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemsLeft* to the default value of 0.

---

## LM\_INSERTITEM

This message inserts an item into a list box control.

### Parameters

**param1**

**sItemIndex** (SHORT)

Item index.

LIT\_END

Add the item to the end of the list.

LIT\_SORTASCENDING

Insert the item into the list sorted in ascending order.

LIT\_SORTDESCENDING

Insert the item into the list sorted in descending order.

Other

Insert the item into the list at the offset specified by this zero-based index.

**param2**

**pszItemText** (PSZ)

Item text.

This points to a string containing the item text.

## Returns

### ***sIndexInserted*** (SHORT)

Index of inserted item.

**LIT\_MEMERROR** The list box control cannot allocate space to insert the list item in the list.

**LIT\_ERROR** An error, other than **LIT\_MEMERROR**, occurred.

**Other** The zero-based index of the offset of the item within the list.

## Remarks

The list box control window procedure responds to this message by inserting the item text identified by the *pszItemText* parameter into the position in the list specified by the *sItemIndex* parameter.

The sorting sequence used is that defined by the `WinCompareStrings` function.

The list box control sets *sIndexInserted* to the zero-based index of the offset of the item within the list.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sIndexInserted* to the default value of 0.

---

## LM\_INSERTMULTITEMS

This message inserts one or more items into a list box.

## Parameters

**param1**

**pListBoxInfo** (PLBOXINFO)

Pointer to a structure containing list box information.

**param2**

**papszText** (PSZ \*)

Pointer to an array of pointers to text strings.

This parameter is a pointer to an array of pointers to zero-terminated strings. The array must contain at least *ullItemCount* items. (*ullItemCount* is a field in `LBOXINFO`.)

If this parameter is set to `NULL`, a *ullItemCount* number of empty items are inserted into the list. This is useful for ownerdraw listboxes that do not make use of text strings.

## Returns

### **ICount** (LONG)

Number of items successfully inserted into the list.

If the number of items is not the same as *ulItemCount*, an error has occurred.

## Remarks

LM\_INSERTMULTITEMS inserts multiple items into a list box at one time, up to 32768 items.

If either LIT\_SORTASCENDING or LIT\_SORTDESCENDING is specified in the *lItemIndex* field of LBOXINFO, then the complete list is sorted after the items have been inserted. If items are being added using several LM\_INSERTMULTITEMS messages, it is faster to specify LIT\_END for all the insert messages except the last one, and then set one of the sort flags to sort the entire list after the last set of items have been inserted.

The sorting sequence is the same as that defined for WinCompareStrings.

WM\_MEASUREITEM (in List Boxes) is sent to the owner of an ownerdraw list box for every item inserted into the list box.

## Default Processing

The default message procedure sets *ICount* to zero.

---

## LM\_QUERYITEMCOUNT

This message returns a count of the number of items in the list box control.

## Parameters

### **param1**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **sItemCount** (SHORT)

Item count.

## Remarks

The list box control window procedure responds to this message by setting *sItemCount* to the number of items in the list.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemCount* to the default value of 0.

---

## LM\_QUERYITEMHANDLE

This message returns the handle of the indexed item of the list box control.

### Parameters

**param1**

**sItemIndex** (SHORT)

Item index.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulItem** (ULONG)

Item handle.

0        The indexed item does not exist.

Other    Item handle.

### Remarks

The meaning of the item handle is defined by the application. It may, for example, be a pointer to an application defined data structure.

Item handles are initialized to `NULLHANDLE` when an item is created. The list box control window procedure responds to this message by setting *ulItem* to the handle of the item whose index is specified by *sItemIndex*.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ulItem* to the default value of `NULLHANDLE`.

The item handle is initialized to `NULLHANDLE`.



---

## LM\_QUERYITEMTEXT

This message returns the text of the specified list box item.

### Parameters

**param1**

**sItemIndex** (SHORT)

Item index.

**sMaxCount** (SHORT)

Maximum count.

0 No text is copied.

Other Copy the item text as a null-terminated string, but limit the number of characters copied, including the null termination character, to this value.

**param2**

**pszItemText** (PSZ)

Buffer into which the item text is to be copied.

This points to a string (character) buffer.

### Returns

**sTextLength** (SHORT)

Length of item text.

The length of the text string, excluding the null termination character.

### Remarks

The list box control window procedure responds to this message by copying up to *sMaxCount* characters, as a null-terminated string, from the text of the item specified by *sItemIndex* into the buffer identified by *pszItemText*.

The length of the item text can be determined by using the LM\_QUERYITEMTEXTLENGTH message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sTextLength* to the default value of 0.

---

## LM\_QUERYITEMTEXTLENGTH

This message returns the length of the text of the specified list box item.

### Parameters

**param1**

**sItemIndex** (SHORT)

Item index.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**sTextLength** (SHORT)

Length of item text.

The length of the text string, excluding the null termination character.

**LIT\_ERROR** Error occurred. For example, the item specified by its index does not exist.

**Other** Length of item text.

### Remarks

The list box control window procedure responds to this message by setting *sTextLength* to the length in characters of the text of the item specified by *sItemIndex*.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than set *sTextLength* to the default value of 0.

---

## LM\_QUERYSELECTION

This message is used to enumerate the selected item, or items, in a list box.

### Parameters

**param1**

**sItemStart** (SHORT)

Index of the start item.

If the list box allows multiple selected items, that is, if it has a style of **LS\_MULTIPLESEL**, then this parameter indicates the index of the item from which the search for the next selected item is to begin. Therefore, to get all the selected

items of the list, this message is sent repeatedly, each time setting this parameter to the index of the item returned by the previous usage of this message.

If this parameter is set to `LIT_CURSOR` the index of the item in the list box which currently has the cursor is returned.

If the list box only allows a single selection, this parameter is ignored.

- |                         |  |
|-------------------------|--|
| <code>LIT_CURSOR</code> | Return the index of the item in the list box which currently has the cursor. |
| <code>LIT_FIRST</code>  | Start the search at the first item.  |
| Other                   | Start the search after the item specified by this index.                     |

## param2

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **sItemSelected** (SHORT)

Index of the selected item.

`LIT_NONE` No selected item.

For a single selection list box, this implies that there is no selected item in the list box. For a multiple selection list box, this implies that there is no selected item in the list box whose index is higher than the index specified by the *sItemStart* parameter.

Other Index of selected item. For a single selection list box, this is the index of the only selected item in the list box. For a multiple selection list box, this is the index of the next selected item in the list box whose index is higher than the index specified by the *sItemStart* parameter.

If *sItemStart* is set to `LIT_CURSOR`, the index of the list-box item which currently has the cursor is returned.

## Remarks

The list box control window procedure responds to this message by returning in *sItemSelected* the zero-based index of the selected item or next selected item after *sItemStart*, if any.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than set *sItemSelected* to the default value of 0.

---

## LM\_QUERYTOPINDEX

This message obtains the index of the item currently at the top of the list box.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**sItemTop** (SHORT)

Index of the item currently at the top of the list box:

LIT\_NONE No items in the list box

Other Index of the item currently at the top of the list box.

### Remarks

The list box control window procedure responds to this message by returning in *sItemTop* the zero-based index of the item currently at the top of the list box.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemTop* to the default value of 0.

---

## LM\_SEARCHSTRING

This message returns the index of the list box item whose text matches the string.

### Parameters

#### param1

**uscmd** (USHORT)

Command.

Defines the criteria by which the string specified by the *pszSearchString* parameter is to be compared with the text of the items, to determine the index of the first matching item.

These values can be combined using the logical-OR operator:

|                   |   |
|-------------------|---|
| LSS_CASESENSITIVE | Matching occurs if the item contains the characters specified by the <i>pszSearchString</i> parameter exactly.<br><i>This value is mandatory.</i>   |
| LSS_PREFIX        | Matching occurs if the leading characters of the item contain the characters specified by the <i>pszSearchString</i> parameter.<br><br>If this value is specified, LSS_SUBSTRING must not be specified. |
| LSS_SUBSTRING     | Matching occurs if the item contains a substring of the characters specified by the <i>pszSearchString</i> parameter.<br><br>If this value is specified, LSS_PREFIX must not be specified.              |

**sItemStart (SHORT)**

Index of the start item.

- LIT\_FIRST Start the search at the first item.
- Other Start the search after the item specified by this index.

**param2**

**pszSearchString (PSZ)**

Search string.

This points to the string to search for.

**Returns**

**sItemMatched (SHORT)**

Index item whose text matches the string.

- LIT\_ERROR Error occurred
- LIT\_NONE No item found
- Other Index item whose text matches the string.

**Remarks**

The list box control window procedure responds to this message by setting *sItemMatched* to the index of the next item whose text matches the string specified by *pszSearchString*.

All the items of the list are searched until a match is found, that is, the search wraps from the end to the start of the list.

**Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemMatched* to the default value of 0.

---

## LM\_SELECTITEM

This message is used to set the selection state of an item in a list box.

### Parameters

#### param1

##### **sItemIndex** (SHORT)

Index of the item to be selected or deselected:

LIT\_NONE All items are to be deselected

Other Index of the item to be selected or deselected.

#### param2

##### **usselect** (USHORT)

Select flag.

(Ignored if *sItemIndex* is set to LIT\_NONE).

TRUE The item is selected. If the control is a single selection list box (that is, it does not have the style of LS\_MULTIPLESEL), any previously selected item is deselected.

FALSE The item is deselected.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred. For example, when the item does not exist in the list box, or when an item that is not selected is deselected.

### Remarks

The list box control window procedure responds to this message by setting the selection state, as indicated by *usselect*, of the item whose index is specified in *sItemIndex*.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## LM\_SETITEMHANDLE

This message sets the handle of the specified list box item.

### Parameters

#### param1

**sItemIndex** (SHORT)  
Item index.

#### param2

**ulItemHandle** (ULONG)  
Item handle.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Remarks

The meaning of the item handle is defined by the application. It may, for example, be a pointer to an application defined data structure.

Item handles are initialized to NULLHANDLE when an item is created.

The list box control window procedure responds to this message by setting the handle of the item whose index is specified by *sItemIndex* to the value specified by *ulItemHandle*.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## LM\_SETITEMHEIGHT

This message sets the height of the items in a list box.

### Parameters

**param1**

**flNewHeight** (ULONG)

Height of items in list box.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful operation

FALSE Error occurred.

### Remarks

The list box control window procedure responds to this message by setting the height of the items in a list box to that specified by *flNewHeight*.

This message does *not* send a WM\_MEASUREITEM message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## LM\_SETITEMTEXT

This message sets the text into the specified list box item.

### Parameters

**param1**

**sItemIndex** (SHORT)

Item index.



**param2**

**pszItemText (PSZ)**

Item text.

This points to a string containing the text to set the list-box item to.

### Returns

**rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Remarks

The list box control window procedure responds to this message by copying the text identified by the *pszItemText* parameter into the item in the list specified by the *sItemIndex* parameter.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## LM\_SETITEMWIDTH

This message sets the width of the items in a list box.

### Parameters

**param1**

**INewWidth (ULONG)**

Width of items in list box.

**param2**

**reserved (ULONG)**

Reserved value, should be 0.

### Returns

**rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The list box control window procedure responds to this message by setting the width of the items in a list box to that specified by *INewWidth*.

**Note:** Only list boxes with the LS\_HORIZSCROLL style set will respond to this message.

This message does *not* send a WM\_MEASUREITEM message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## LM\_SETTOPINDEX

This message is used to scroll a particular item to the top of the list box.

### Parameters

**param1**

**sItemIndex** (SHORT)

Index of the item to be made top.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The list box control window procedure responds to this message by scrolling the item whose index is identified by *sItemIndex* to the top of the list box.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## WM\_CHAR (in List Boxes)

For the cause of this message, see "WM\_CHAR" on page 10-32.

For a description of the parameters, see "WM\_CHAR" on page 10-32.

### Remarks

The list box control window procedure responds to this message by sending it to its owner if it has not processed the key stroke. This is the most common means by which the input focus is switched around the various controls in a dialog box.

The key strokes processed by a list box control are:

- Down Arrow** Moves the selection down one item, scrolling the list box by one item, if necessary, to make the next item visible. When the selection reaches the bottom, the Down Arrow has no effect.
- Up Arrow** Moves the selection up one item, scrolling the list box by one item, if necessary, to make the previous item visible. When the selection reaches the top, the Up Arrow has no effect.
- Page Down** Moves the selection down one page, scrolling the list box by the number of items visible in the list box.
- For example, if the list box displays seven items and item 1 is selected and positioned at the top of the list box, pressing the Page Down key causes item 8 to be selected and displayed at the top of the list box. Pressing Page Down when the last item is selected has no effect.
- Page Up** Moves the selection up one page, scrolling the list box by the number of items visible in the list box.
- For example, if the list box displays seven items and item 8 is selected and positioned at the top of the list box, pressing the Page Up key causes item 1 to be selected and displayed at the top of the list box. Pressing the Page Up key when the first item is selected has no effect.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE

### Related Messages

- WM\_CHAR

---

## **WM\_QUERYCONVERTPOS (in List Boxes)**

For the cause of this message, see “WM\_QUERYCONVERTPOS” on page 10-72.

For a description of the parameters, see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Remarks**

The list box control window procedure returns QCP\_NOCONVERT.

### **Default Processing**

For the default window procedure processing of this message see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Related Messages**

- WM\_QUERYCONVERTPOS

---

## **WM\_QUERYWINDOWPARAMS (in List Boxes)**

Occurs when an application queries the list box control window parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### **Remarks**

The list box control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to 0 and sets *rc* to FALSE.

### **Related Messages**

- WM\_QUERYWINDOWPARAMS

---

## **WM\_SETWINDOWPARAMS (in List Boxes)**

This message occurs when an application sets or changes the list box control window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### **Remarks**

The list box control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure takes no action on this message, other than to set **result** to FALSE.

## **Related Messages**

- `WM_SETWINDOWPARAMS`

---

## Chapter 15. Menu Control Window Processing

This system-provided window procedure processes the actions on a menu control (WC\_MENU).

### Purpose

A menu control is a child or pull-down window that contains a list of selection items. These items can be represented by text strings, separators, bit maps or menu buttons. Menu templates can be loaded as resources and the menu can be created automatically when the parent window is created. The application can build the menu dynamically by sending MM\_INSERTITEM messages. An application can change a menu by sending messages to it.

Menus enable the operator to select one of the items in the list, using the pointing device or the keyboard. When a selection is made, the menu parent is notified by posting a WM\_COMMAND, WM\_SYSCOMMAND, or WM\_HELP message and a unique identifier representing the operator's selection.

Menus automatically resize themselves when items are added and removed. Menus are automatically destroyed when their owner is destroyed.

Typically, an application has an action bar menu and several submenus. The action bar is normally visible, and is a child window in the parent window frame. The submenus are normally hidden and become visible when selections are made on the action bar.

---

### Menu Control Styles

These menu control styles are available:

#### MS\_ACTIONBAR

The items in the list are displayed side-by-side. This style is used to implement a top level menu. Menus that do not have this style are displayed in one or more columns and are submenus associated with an action bar.

All menu controls have styles CS\_SYNCPAINT and CS\_PARENTCLIP.

#### MS\_CONDITIONALCASCADE

This style is used to specify that the items in this list are a conditional cascade menu. Conditional cascade menus act like normal cascade menus with the exception that the cascade does not automatically open when the user selects it. To open the conditional cascade menu, the mini-pushbutton on the menu item must be selected. If the menu is selected without opening the cascade, the default item in the cascade is selected. The default action on the cascade is identified by a check mark.

**MS\_TITLEBUTTON**

Used to identify menus that can be used as buttons in the title bar. Can only be used with MS\_ACTIONBAR.

This style causes the menu to be drawn using the CUA colors specified for the title bar rather than the action bar.

**MS\_VERTICALFLIP**

Normally, pull-down menus (the default, without the MS\_VERTICALFLIP style) are displayed below their associated action bar item. If there is not room on the screen to display the entire pull-down in this manner, and if there is room to display the pull-down above the action bar, it is displayed above the action bar. Pull-down menus with the MS\_VERTICALFLIP style are flipped vertically. That is, they are displayed above the menu if possible, otherwise below it. The vertical flip style must be set explicitly by the application when the window is minimized, and must be reset when it is restored.

If an application action bar contains this style, the style is applied to all pull-down menus belonging to the action bar (the style does not directly affect the display of the action bar). This provides a convenient means for the application to flip the appearance of all pull-down menus.

---

## Menu Item Styles

These menu item styles are available:

**MIS\_SUBMENU**

The item is a submenu. When the user selects this type of item, a submenu is displayed from which the user must make further selection. Items that are not submenu items are command items.

**MIS\_SEPARATOR**

The display object is a horizontal dividing line. This type of item can only be used in pull-down menus. This type of item cannot be enabled, checked, disabled, highlighted, or selected by the user. The functional object is NULL when this style is specified.

**MIS\_BITMAP**

The display object is a bit map.

**MIS\_TEXT**

The display object is a text string.

**MIS\_BUTTONSEPARATOR**

The item is a menu button. Any menu can have zero, one, or two items of this type. These are the last items in a menu and are automatically displayed after a separator bar. The user cannot move the cursor to these items, but can select them with the pointing device or with the appropriate key.

**MIS\_BREAK**

The item begins a new row or column.

|                           |  |
|---------------------------|--|
| <b>MIS_BREAKSEPARATOR</b> | Same as MIS_BREAK, except that it draws a separator between rows or columns of a pull-down menu. This style can only be used within a submenu.                   |
| <b>MIS_SYSCOMMAND</b>     | If this item is selected, the menu notifies the owner by posting a WM_SYSCOMMAND message rather than a WM_COMMAND message.                                       |
| <b>MIS_OWNERDRAW</b>      | Items with this style are drawn by the owner. WM_DRAWITEM and WM_MEASUREITEM notification messages are sent to the owner to draw the item or determine its size. |
| <b>MIS_HELP</b>           | If the item is selected, the menu notifies the owner by posting a WM_HELP message rather than a WM_COMMAND message.  |
| <b>MIS_STATIC</b>         | This type of item exists for information purposes only. It cannot be selected with the pointing device or keyboard.  |

---

## Menu Item Attributes

Applications can get and set the state of these attributes by sending MM\_QUERYITEMATTR and MM\_SETITEMATTR messages.

These menu item attributes are available:

|                      |  |
|----------------------|--|
| <b>MIA_HILITED</b>   | The state of this attribute is TRUE, if and only if, the item is selected.   |
| <b>MIA_CHECKED</b>   | If this attribute is TRUE a check mark appears next to the item.   |
| <b>MIA_DISABLED</b>  | This attribute is TRUE if the item is disabled and cannot be selected. The item is drawn in a disabled state.  |
| <b>MIA_FRAMED</b>    | If this attribute is TRUE a frame is drawn around the item.  |
| <b>MIA_NODISMISS</b> | If this item is selected, the pull-down menu containing this item should not be hidden before notifying the application window of the selection. A menu with this attribute is not hidden until such time as the application or user explicitly does so, for example by selecting either another menu on the action bar or by pressing the escape key. |

---

## Default Colors

The following system colors are used when the system draws button controls:

SYSCLR\_WINDOWFRAME  
 SYSCLR\_BUTTONDARK  
 SYSCLR\_BUTTONLIGHT  
 SYSCLR\_SHADOW  
 SYSCLR\_TITLEBOTTOM  
 SYSCLR\_DIALOGBACKGROUND



Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_FOREGROUND  
PP\_HILITEFOREGROUND  
PP\_BORDER  
PP\_DISABLEDFOREGROUND

---

## Menu Control Notification Messages

These messages are initiated by the menu control window procedure to notify its owner of significant events.

---

### WM\_COMMAND (in Menu Controls)

For the cause of this message, see “WM\_COMMAND” on page 10-37.

For a description of the parameters, see “WM\_COMMAND” on page 10-37.

The menu control window procedure sets *uscmd* to the menu-item identity.

#### Remarks

The menu control window procedure generates this message if the WM\_MENUSELECT (in Menu Controls) message returns a *rc* of TRUE. when an item is selected that does not have the style of MIS\_SYSCOMMAND or MIS\_HELP. The menu control window procedure posts the message to the queue of the window owner.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_COMMAND

---

### WM\_DRAWITEM (in Menu Controls)

This notification is sent to the owner of a menu control each time an item is to be drawn.

#### Parameters

**param1**

**idMenu** (USHORT)

Window identifier.

The window identity of the menu control sending this notification message.

**param2**

**pOwnerItem** (OWNERITEM)

Owner-item structure.

This points to an owner-item structure; see “OWNERITEM” on page A-136.

## Returns

**rc** (BOOL)

Item-drawn indicator.

- TRUE The owner draws the item, and so the menu control does not draw it.  
FALSE If the item contains text and the owner does not draw the item, the owner returns this value and the menu control draws the item.

## Remarks

The menu control window procedure only draws items that are represented by text strings and emphasizes selected items by inverting them.

If an application uses menu controls containing items that are not represented by text strings, or requires that the emphasized state of an item is to be drawn in a special manner, then the menu control must specify the style `MIS_OWNERDRAW` and those items must be drawn by the owner.

The menu control window procedure generates this message and sends it to its owner, informing the owner that an item is to be drawn, offering the owner the opportunity to draw that item, and to indicate that either the item has been drawn, or that the menu control is to draw it.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

## Related Messages

- `WM_DRAWITEM`

---

## WM\_HELP (in Menu Controls)

For the cause of this message, see “WM\_HELP” on page 10-49.

For a description of the parameters, see “WM\_HELP” on page 10-49.

The menu control window procedure sets *uscmd* to the menu-item identity.

## Remarks

This message is identical to a `WM_COMMAND` message, but implies that the application should respond to this message by displaying help information.

The menu control window procedure generates this message and posts it to the queue of its owner when an item is selected that has the style of `MIS_HELP`, but only if `WM_MENUSELECT` (in Menu Controls) returns a *rc* of TRUE.

## Default Processing

The default window procedure sends this message to the parent window, if it exists and is not the desktop. Otherwise, it sets *ulReserved* to 0.

## Related Messages

- WM\_HELP

---

## WM\_INITMENU (in Menu Controls)

For the cause of this message, see “WM\_INITMENU” on page 10-53.

For a description of the parameters, see “WM\_INITMENU” on page 10-53.

### Remarks

This message offers the owner the opportunity to perform some initialization on the menu items before they are presented.

The menu control window procedure generates this message and sends it to its owner, informing the owner of the event.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_INITMENU

---

## WM\_MEASUREITEM (in Menu Controls)

This notification is sent to the owner of a menu control to establish the height for an item in that control.

### Parameters

**param1**

**sMenu** (SHORT)

Menu identifier.

**param2**

**pOwnerItem** (OWNERITEM)

Owner-item structure.

This points to an OWNERITEM structure.

## Returns

### **sHeight** (SHORT)

Height of item.

## Remarks

This message is only sent at the time the menu control is created. When the owner receives this message, it must calculate and return the height of an item to the control.

All items in a menu must have the same height, and that must be greater than or equal to the height of the current font.

In particular, this notification is sent to the owner of a menu that has a style of `MIS_OWNERDRAW`, to offer the owner an opportunity to establish the height of an item that accommodates any special requirements for the drawing of items in that menu.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sHeight* to the default value of 0.

## Related Messages

- `WM_MEASUREITEM`

---

## **WM\_MENUEND** (in Menu Controls)

For the cause of this message, see “`WM_MENUEND`” on page 10-56.

For a description of the parameters, see “`WM_MENUEND`” on page 10-56.

## Remarks

The menu control window procedure generates this message and sends it to its owner, informing the owner of this event.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- `WM_MENUEND`

---

## **WM\_MENUSELECT** (in Menu Controls)

For the cause of this message, see “`WM_MENUSELECT`” on page 10-57.

For a description of the parameters, see “`WM_MENUSELECT`” on page 10-57.

### **Remarks**

The menu control window procedure generates this message and sends it to its owner, informing the owner of this event.

When the message is returned from its owner, menu control acts on *rc* as appropriate.

It must not be posted to the menu control.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to TRUE.

### **Related Messages**

- WM\_MENUSELECT

---

## **WM\_NEXTMENU (in Menu Controls)**

For the cause of this message, see “WM\_NEXTMENU” on page 10-63.

For a description of the parameters, see “WM\_NEXTMENU” on page 10-63.

### **Remarks**

The menu control generates this message and sends it to its owner, informing the owner of this event.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *hwndNewMenu* to NULLHANDLE.

### **Related Messages**

- WM\_NEXTMENU

---

## Menu Control Window Messages

This section describes the menu control window procedure actions on receiving the following messages.

---

### MM\_DELETEITEM

This message deletes a menu item.

#### Parameters

##### param1

###### **usitem** (USHORT)

Item identifier.

###### **usincludesubmenus** (USHORT)

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and delete it.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

##### param2

###### **ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

##### **sItemsLeft** (SHORT)

Number remaining.

The number of items in the menu after the item is deleted.

#### Remarks

The menu control window procedure responds to this message by deleting the identified item from the menu or its submenus.

**Note:** It must be sent, not posted, to the menu control.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemsLeft* to the default value of 0.

---

## MM\_ENDMENUODE

This message is sent to a menu control to terminate menu selection.

### Parameters

#### param1

##### **usdismiss** (USHORT)

Dismiss menu indicator.

TRUE Dismiss the submenu or subdialog window

FALSE Do not dismiss the submenu or subdialog window.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The menu control window procedure responds to this message by terminating menu selection.

**Note:** It must be sent, not posted, to the menu control.

### Default Processing

The default window procedure does not expect to receive this message and, therefore, takes no action on it, other than to set *ulReserved* to the default value of 0.

---

## MM\_INSERTITEM

This message inserts a menu item into a menu.

### Parameters

#### param1

##### **pmenuitem** (PMENUITEM)

Menu-item data structure.

This points to a MENUITEM structure.



## param2

### **pszItemText** (PSZ)

Item text.

This points to a string containing the text to be inserted.

## Returns

### **sindexInserted** (SHORT)

Index of inserted item.

**MIT\_MEMERROR** The menu control cannot allocate space to insert the menu item in the menu.

**MIT\_ERROR** An error other than **MIT\_MEMERROR** occurred.

**Other** The zero-based index of the offset of the item within the menu.

## Remarks

The menu control window procedure responds to this message by inserting the identified item into the menu at the position indicated by the specified **MENUITEM** data structure (contained within the menu-item structure). If the position is **MIT\_END**, the item is added to the end of the menu. If the style of the item includes **MIS\_TEXT**, the text of the item is specified by *pszItemText*

The menu control window procedure sets *sindexInserted* to the zero-based index of the position of the item within the menu.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sindexInserted* to the default value of 0.

---

## **MM\_ISITEMVALID**

This message returns the selectable status of a specified menu item.

## Parameters

### param1

#### **usitem** (USHORT)

Item identifier.

#### **usincludesubmenus** (USHORT)

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

## **param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## **Returns**

**rc** (BOOL)

Selectable indication.

A menu item can be selected and entered under these conditions:

- The item is enabled and, if it is a submenu item, the item in the action bar associated with the submenu is enabled. If the action bar item is not enabled, the user cannot display the submenu.
- The item is enabled, and the submenu is displayed and being tracked with the pointing device or keyboard. It is unlikely, but possible, that the associated action bar is disabled in this instance.

**TRUE** The user can select and enter the specified item.

**FALSE** The user cannot select and enter the specified item.

## **Remarks**

The menu control window procedure responds to this message by setting the return value depending on the selectable status of the specified item.

## **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of **FALSE**.

---

## **MM\_ITEMIDFROMPOSITION**

This message returns the identity of a menu item of a specified index.

## **Parameters**

**param1**

**sltemIndex** (SHORT)

Item index.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**sIdentity** (SHORT)

Item identity.

MIT\_ERROR Error occurred; for example, because *sItemIndex* is not valid.

Other Item identity.

## Remarks

The menu control window procedure responds to this message by setting *sIdentity* to the identity of the item whose position is identified by the index specified in *sItemIndex*.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sIdentity* to the default value of 0.

---

## MM\_ITEMPOSITIONFROMID

This message returns the index of a menu item of a particular identity.

## Parameters

**param1**

**usItem** (USHORT)

Item identifier.

**usIncludesSubmenus** (USHORT)

Include submenus indicator.

TRUE If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier.

FALSE If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **sIndex** (SHORT)

Item index.

|          |                     |
|----------|---------------------|
| MIT_NONE | Item does not exist |
| Other    | Item index.         |

## Remarks

The menu control window procedure responds to this message by setting *sIndex* to the zero-based index of the item identified by *sIndex*.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sIndex* to the default value of MIT\_NONE.

---

## MM\_QUERYDEFAULTITEMID

This message returns the default item id for a conditional cascade menu. For any other type of menu or submenu, this message returns zero.

## Parameters

### param1

**ulReserved** (ULONG)

Reserved value, must be 0.

### param2

**ulReserved** (ULONG)

Reserved value, must be 0.

## Returns

### **ulDefItemID** (ULONG)

Menu id of the default menu item.

## Default Processing

The default window procedure takes no action other than to return 0.

## Related Messages

- WM\_DRAWITEM (in Frame Controls)
- WM\_DRAWITEM (in List Boxes)
- WM\_DRAWITEM (in Menu Controls)

---

## MM\_QUERYITEM

This message returns the definition of the specified menu item.

### Parameters

#### param1

**usitem** (USHORT)

Item identifier.

**usincludesubmenus** (USHORT)

Include submenus flag.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and copy its definition.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

#### param2

**pmenuitem** (PMENUITEM)

Menu-item data structure.

This points to a MENUITEM structure.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Successful completion

**FALSE** Error occurred.

### Remarks

The menu control window procedure responds to this message by copying the item definition specified by *usitem*, from the menu, to the structure specified by *pmenuitem*.

**Note:** This message does not retrieve the text for items with a style of MIS\_TEXT. The item text is obtained by use of the MM\_QUERYITEMTEXT message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## MM\_QUERYITEMATTR

This message returns the attributes of a menu item.

### Parameters

#### param1

**usitem** (USHORT)

Item identity.

**usIncludeSubmenus** (USHORT)

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and return its state.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

#### param2

**usattributemask** (USHORT)

Attribute mask.

### Returns

**usState** (USHORT)

State.

### Remarks

The menu control responds to this message by returning the state of the specified attributes of the identified menu item.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *usState* to the default value of 0.

### Examples

This example sends an MM\_QUERYITEMATTR message to find the state of the 'idCase' menu item. It then toggles the state of the item and sends an MM\_SETITEMATTR message to set the new state.

```
sState = (SHORT) WinSendMsg(hwndMenu, MM_QUERYITEMATTR,  
    MPFROM2SHORT(idCase, TRUE), MPFROMSHORT(MIA_CHECKED));  
sState ^= MIA_CHECKED;  
WinSendMsg(hwndMenu, MM_SETITEMATTR, MPFROM2SHORT(idCase, TRUE),  
    MPFROM2SHORT(MIA_CHECKED, sState));
```

---

## MM\_QUERYITEMCOUNT

This message returns the number of items in the menu.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**sresult** (SHORT)

Item count.

### Remarks

The menu control window procedure responds to this message by returning the count of the number of items in the menu.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sresult* to the default value of 0.

---

## MM\_QUERYITEMRECT

This message returns the bounding rectangle of a menu item.

### Parameters

**param1**

**usitem** (USHORT)

Item identity.

**fincludeSubmenus (BOOL)**

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and return its state.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

**param2****prect (PRECTL)**

Bounding rectangle of the menu item in device coordinates relative to the menu window.

**Returns****rc (BOOL)**

Success indicator.

**TRUE** Specified item was found.

**FALSE** Specified item was not found.

**Remarks**

The menu control responds to this message by returning the bounding rectangle of identified menu item.

**Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of 0 (**FALSE**).

---

**MM\_QUERYITEMTEXT**

This message returns the text of the specified menu item.

**Parameters****param1****usitem (USHORT)**

Item identifier.

**smaxcount (SHORT)**

Maximum count.

Copy the item text as a null-terminated string, but limit the number of characters copied, including the null termination character, to this value, which must be greater than 0.



**param2**

**pszItemText (PSZ)**

Buffer into which the item text is to be copied.

This points to a string (character) buffer.

## Returns

**sTextLength (SHORT)**

Length of item text.

The length of the text string, excluding the null termination character.

0 Error occurred. For example, no item of the specified identity exists or the item has no text. No text is copied.

Other Length of item text.

## Remarks

The menu control window procedure responds to this message by copying up to *smaxcount* characters as a null-terminated string from the text of the item specified by *usitem*, if it has the style MIS\_TEXT, into the buffer specified by *pszItemText*.

The length of the item text can be determined by using the MM\_QUERYITEMTEXTLENGTH message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sTextLength* to the default value of 0.

---

## MM\_QUERYITEMTEXTLENGTH

This message returns the text length of the specified menu item.

## Parameters

**param1**

**usitem (USHORT)**

Item identifier.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

### **sLength** (SHORT)

Length of item text.

The length of the text string, excluding the null termination character.

0 Error occurred. For example, no item of the specified identity exists or the item has no text. No text is copied.

Other Length of item text.

## Remarks

The menu control window procedure responds to this message by returning the length in characters of the text of the identified item, if it has a style of MIS\_TEXT.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sLength* to the default value of 0.

---

## MM\_QUERYSELITEMID

This message returns the identity of the selected menu item.

## Parameters

### **param1**

#### **usReserve** (USHORT)

Reserved value, should be 0.

#### **usincludesubmenus** (USHORT)

Include submenus indicator.

TRUE If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for a selected item with the specified identifier.

FALSE If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for a selected item with the specified identifier.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **sresult** (SHORT)

Selected item identifier.

MID\_ERROR Error occurred

MIT\_NONE No item selected

Other Selected item identifier.

## Remarks

The menu control window procedure responds to this message by returning the identity of the selected item in the menu. Submenus and subdialogs are not searched unless *usincludesubmenus* is set to TRUE.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sresult* to the default value of 0.

---

## MM\_REMOVEITEM

This message removes a menu item.

## Parameters

### **param1**

#### **usitem** (USHORT)

Item identifier.

#### **usincludesubmenus** (USHORT)

Include submenus indicator.

TRUE If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and delete it.

FALSE If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **sitemsLeft** (SHORT)

Count of remaining items.

## Remarks

The menu control window procedure responds to this message by removing the identified item from the menu and setting *sItemsLeft* to the count of items in the menu after the item is deleted.

The difference between this message and MM\_DELETEITEM is that MM\_DELETEITEM destroys any submenu window, and deletes any bit map associated with the item, whereas MM\_REMOVEITEM does not.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *sItemsLeft* to the default value of 0.

---

## MM\_SELECTITEM

This message selects or deselects a menu item.

### Parameters

**param1**

**sitem** (SHORT)

Item identifier.

MIT\_NONE     Deselect all the items in the menu.

Other         Item identifier.

**usincludesubmenus** (USHORT)

Include submenus indicator.

TRUE        If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and select or deselect it.

FALSE       If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

**param2**

**usReserve** (USHORT)

Reserved value, should be 0.

**usdismissed** (USHORT)

Dismissed flag.

TRUE        Dismiss the menu

FALSE       Do not dismiss the menu.

## Returns

**rc** (BOOL)

Success indicator.

TRUE A selection has been made, or *sitem* is MIT\_NONE.

FALSE A selection has not been made, or a deselection has been made, or *sitem* is not MIT\_NONE.

## Remarks

The menu control window procedure responds to this message by setting the selection state of the (sub)menu which contains the specified item to indicate that the item is selected or deselected. If *usincludessubmenus* is set to TRUE, the selection state of the (sub)menu owning the submenu which contains the specified item is also set. This process continues up the menu hierarchy until the top level menu is reached.

If an item is selected, and *usdismissed* is set to TRUE, a WM\_COMMAND, WM\_SYSCOMMAND, or WM\_HELP message, as appropriate, is posted to the owner, and the menu is dismissed.

**Note:** This message must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## MM\_SETDEFAULTITEMID

This message is used to set the default item in a conditional cascade menu.

## Parameters

**param1**

**ulDefitemID** (ULONG)

The menu id of the item to become the new default.

**param2**

**ulReserved** (ULONG)

Reserved value, must be 0.

## Returns

**rc** (BOOL)

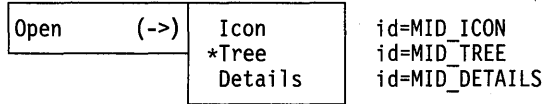
Success of failure indicator.

TRUE The conditional cascade default was set.

FALSE The conditional cascade default was not set.

## Remarks

The default item is the menu-id that will be returned if the main menu option is clicked on.



In the example above, where MID\_TREE is currently the default, if the user clicked on the "Open" option without opening the conditional cascade menu, the menu would send back a notification that MID\_TREE was selected.

## Default Processing

The default window procedure takes no action other than to return 0.

---

## MM\_SETITEM

This message sets the definition of a menu item.

### Parameters

**param1**

**usReserve** (USHORT)

Reserved value, should be 0.

**usincludesubmenus** (USHORT)

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and set its definition.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

**param2**

**pmenuitem** (PMENUITEM)

Menu-item data structure.

This points to a MENUITEM structure.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Successful completion

**FALSE** Error occurred.

## Remarks

The menu control window procedure responds to this message by using the specified structure to update the definition of the identified menu item.

The *iPosition* field of the structure specified by *pmenuitem* is ignored, as the position of the item cannot be changed by use of this message.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## MM\_SETITEMATTR

This message sets the attributes of a menu item.

### Parameters

**param1**

**usitem** (USHORT)

Item identifier.

**usincludesubmenus** (USHORT)

Include submenus indicator.

**TRUE** If the menu does not have an item with the specified identifier, search the submenus and subdialogs of the menu for an item with the specified identifier and set its attributes.

**FALSE** If the menu does not have an item with the specified identifier, do not search the submenus and subdialogs of the menu for an item with the specified identifier.

**param2**

**usattributemask** (USHORT)

Attribute mask.

**usattributedata** (USHORT)

Attribute data.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Successful completion

**FALSE** Error occurred.

## Remarks

The menu control window procedure responds to this message by setting the state of the specified attributes for the identified item.

**Note:** It must be sent, not posted, to the menu control.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

## Examples

This example sends an MM\_SETITEMATTR message to set the IDM\_LARGE menu item's state to checked, and then sends another MM\_SETITEMATTR message to set the IDM\_MEDIUM menu item's state to unchecked.

```
WinSendMessage(hwndActionBar, MM_SETITEMATTR,  
    MPFROM2SHORT(IDM_LARGE, TRUE),  
    MPFROM2SHORT(MIA_CHECKED, MIA_CHECKED));  
WinSendMessage(hwndActionBar, MM_SETITEMATTR,  
    MPFROM2SHORT(IDM_MEDIUM, TRUE),  
    MPFROM2SHORT(MIA_CHECKED, FALSE));
```

---

## MM\_SETITEMHANDLE

This message sets the handle of a menu item.

### Parameters

**param1**

**usitem** (USHORT)  
Item index.

**param2**

**ulitemhandle** (ULONG)  
Item handle.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion  
FALSE Error occurred.



## Remarks

The menu control window procedure responds to this message by setting the handle of the indexed menu item.

This is used to set a handle for menu items that have a style of `MIS_BITMAP` or `MIS_OWNERDRAW`.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set `rc` to the default value of `FALSE`.

---

## MM\_SETITEMTEXT

This message sets the text of a menu item.

### Parameters

**param1**

**usitem** (USHORT)  
Item identifier.

**param2**

**pszItemText** (PSZ)  
Item text.

This points to a string containing the text to set the menu item to.

### Returns

**rc** (BOOL)

Success indicator.

TRUE    Successful completion  
FALSE   Error occurred.

### Remarks

The menu control responds to this message by setting the text of the identified item, if it has a style of `MIS_TEXT`, using the specified null-terminated string.

**Note:** It must be sent, not posted, to the menu control.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set `rc` to the default value of `FALSE`.

---

## MM\_STARTMENU MODE

This message is used to begin menu selection.

### Parameters

#### param1

##### usshowsubmenu (USHORT)

Show submenu flag.

- TRUE Show the submenu (pull-down menu) of the selected action bar item when the menu enters selection mode. If the action bar is not visible, the submenu is shown, otherwise it is not shown. If the item selected does not have a submenu, this parameter is ignored.
- FALSE Do not show the submenu (pull-down menu) of the selected action bar item when the menu enters selection mode.

##### usresumemenu (USHORT)

Resume menu mode flag.

- TRUE Resume the user interaction with the menu from where it left off. The menu is assumed to have been used previously and left without dismissing one of the submenus, and therefore is resumed in that submenu.
- FALSE Begin user interaction with the menu from the action bar, subject to the value of the *usshowsubmenu* parameter.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

- TRUE Successful completion
- FALSE Error occurred.

### Remarks

It is posted to the menu when the operator presses the menu key.

**Note:** It must be posted, not sent, to the menu control.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## **WM\_QUERYCONVERTPOS (in Menu Controls)**

For the cause of this message, see “WM\_QUERYCONVERTPOS” on page 10-72.

For a description of the parameters, see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Remarks**

The menu control window procedure returns QCP\_NOCONVERT.

### **Default Processing**

For the default window procedure processing of this message see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Related Messages**

- WM\_QUERYCONVERTPOS

---

## **WM\_QUERYWINDOWPARAMS (in Menu Controls)**

Occurs when an application queries the menu control window procedure parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### **Remarks**

The menu control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to 0 and sets *rc* to FALSE.

### **Related Messages**

- WM\_QUERYWINDOWPARAMS

---

## **WM\_SETWINDOWPARAMS (in Menu Controls)**

This message occurs when an application sets or changes the menu control window procedure parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### **Remarks**

The menu control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_SETWINDOWPARAMS

---

## WM\_SYSCOMMAND

For the cause of this message, see "WM\_SYSCOMMAND" on page 10-91.

For a description of the parameters, see "WM\_SYSCOMMAND" on page 10-91.

The menu control window procedure sets *uscmd* to the menu-item identity.

### Remarks

The menu control window procedure generates this message and posts it to the queue of its owner, when an item is selected that has the style of MIS\_SYSCOMMAND, but only if the WM\_MENUSELECT (in Menu Controls) message returns a *rc* of TRUE.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.



---

## Chapter 16. Multi-Line Entry Field Control Window Processing

This system-provided window procedure processes the actions on a multi-line entry field control (WC\_MLE).

### Purpose

A multi-line entry field control is a rectangular window that displays multiple lines of text that the operator can edit. When it has the focus, the cursor marks the current **insertion** or **replacement** point.

### How to Use

The text is displayed within a rectangular window. Scroll bars appear if requested.

On all four sides of the text within the window there exists a thin margin area. This margin remains drawn in the window's background color, and characters are never drawn into this margin. Mouse events that occur in the margin are processed differently from mouse events that occur in the text area. The margin should be large enough to be easily clicked on, but not so large as to take up a large quantity of screen space. It is suggested, but not required, that the left and right margins be half the average character width of the system font, and that the top and bottom margins be half the maximum baseline extent of the system font.

Text is defined as a stream of characters, with hard line-break characters in the text. Between any two bytes in the text stream, and at either end of the document, there is an insertion point. Note that in a DBCS environment, it is possible to have an insertion point in the middle of a DBCS character. If such an insertion point is specified in a function, the function will either round the insertion point in a sensible way, or the function will fail with an error code indicating the problem.

The text always contains a selection region, defined by an anchor point and a cursor point. The anchor and cursor points are insertion points. If the MLE window has the focus, the text between these two points is drawn highlighted and the cursor point is indicated by a flashing text cursor. The selection region can be affected by some import/export operations.

The cursor point and the anchor point define the range of the selection. These two points are often the same, in which case no text is selected and only a text cursor (but no highlighting) is displayed. A user can use SHIFT+cursor movement combinations to extend the selection, which leaves the anchor point alone, and moves the cursor point to a new position in the document.

The MLE has three modes:

**READ-ONLY**      The keyboard user interface disallows any operations that would change the content of the text, although applications using the MLE can still change the text contents. The application can query this mode, in order that it can disallow application-specific operations.

**WORD-WRAP** When this mode is in effect, soft line-breaks are inserted into the text at word boundaries so that the user need not scroll the display horizontally to see all the text. When this mode is off, text is allowed to trail off the right-hand edge of the window.

**INSERT/OVERTYPE** This mode determines whether keystrokes are inserted into the text, or whether they overtype existing text. Unlike the other two modes, this mode is maintained by the system. The MLE must merely be aware of the system mode.

**Notes:**

1. The MLE is intended for text under 4Kb in size. Performance will be fast for text up to 32KB in size. Text greater than this will be supported but performance may not be acceptable.
2. In this chapter 'CR' denotes carriage-return, and 'LF' denotes line-feed.

---

## Multi-Line Entry Field Control Styles

These multi-line entry field control styles are available:

|                        |  |
|------------------------|--|
| <b>MLS_BORDER</b>      | A thin border is drawn around the multi-line entry field window.   |
| <b>MLS_READONLY</b>    | The multi-line entry field is initially in read-only mode.   |
| <b>MLS_WORDWRAP</b>    | The multi-line entry field initially word-wraps text.  |
| <b>MLS_HSCROLL</b>     | The multi-line entry field displays and handles a horizontal scroll bar.                                   |
| <b>MLS_VSCROLL</b>     | The multi-line entry field displays and handles a vertical scroll bar.                                     |
| <b>MLS_IGNORETAB</b>   | The multi-line entry field ignores tab key strokes. It passes the appropriate WM_CHAR to its owner window. |
| <b>MLS_DISABLEUNDO</b> | The multi-line entry field will not allow undo actions.  |

---

## Multi-Line Entry Field Control Data

See "MLECTLDATA" on page A-127.

---

## Multi-Line Entry Field Control Notification Messages

This message is initiated by the multi-line entry field window procedure to notify its owner of significant events.

---

### WM\_CONTROL (in Multiline Entry Fields)

For the cause of this message, see "WM\_CONTROL" on page 10-39.

#### Parameters

##### param1

##### usid (USHORT)

Control window identity.

##### usnotifycode (USHORT)

Notify code.

##### MLN\_TEXTOVERFLOW

A key stroke causes the amount of text to exceed the limit on the number of bytes of data (refer to MLM\_SETTEXTLIMIT). The parameter contains the number of bytes of data which would not fit within the current text limit. For character key strokes this can be 1 or 2 (DBCS). For Shift+Ins (paste) it can be any amount up to the paste limit.

The default *rc* of FALSE causes the default error handling, which is to ignore the key stroke, and beep.

An *rc* of TRUE implies that corrective action has been taken (such as deleting existing text or raising the limit) and the WM\_CHAR (in Multiline Entry Fields) should be reprocessed as if just entered.

##### MLN\_PIXHORZOVERFLOW

A key stroke causes the size of the display bit map to exceed the horizontal limit of the format rectangle (refer to MLM\_SETFORMATRECT). The parameter contains the number of pels that would not fit within the current text limit.

The default *rc* of FALSE causes the default error handling, which is to ignore the key stroke, and beep.

An *rc* of TRUE implies that corrective action has been taken (such as changing to a smaller font or raising the limit) and the WM\_CHAR (in Multiline Entry Fields) should be reprocessed as if just entered.



|                     |   |
|---------------------|---|
| MLN_PIXVERTOVERFLOW | <p>A key stroke causes the size of the display bit map to exceed the vertical limit of the format rectangle (refer to MLM_SETFORMATRECT). The parameter contains the number of pels that would not fit within the current text limit.</p> <p>The default <i>rc</i> of FALSE causes the default error handling, which is to ignore the key stroke, and beep.</p> <p>An <i>rc</i> of TRUE implies that corrective action has been taken (such as changing to a smaller font or raising the limit) and the WM_CHAR (in Multiline Entry Fields) should be reprocessed as if just entered.</p> |
| MLN_OVERFLOW        | <p>An action other than entry of a key stroke causes a condition involving the text limit or format rectangle limit, such that either the limit becomes inadequate to contain the text or the text exceeds the limit.</p> <p>This can be caused by:</p> <ul style="list-style-type: none"> <li>MLM_SETWRAP</li> <li>MLM_SETTABSTOP</li> <li>MLM_SETFONT</li> <li>MLM_IMPORT</li> <li>MLM_PASTE</li> <li>MLM_CUT</li> <li>MLM_UNDO</li> <li>MLM_DELETE</li> <li>WM_SIZE.</li> </ul>  |
| MLN_HSCROLL         | <p>Indicates that the MLE has completed a scrolling calculation and is about to update the display accordingly. All queries return values as if the scrolling were complete. However, no scrolling action is visible on the user interface.</p>   |
| MLN_VSCROLL         | <p>Indicates that the MLE has completed a scrolling calculation and is about to update the display accordingly. All queries return values as if the scrolling were complete. However, no scrolling action is visible on the user interface.</p>   |
| MLN_CHANGE          | <p>Signals that the text has changed. This notification is sent whenever any text change occurs.</p>  |
| MLN_UNDOOVERFLOW    | <p>Signals that the text change operation, which could normally be undone, cannot be undone because the amount of text involved exceeds the undo capability. This includes text entry, deletion, cutting, and pasting.</p>  |

|                 |  |
|-----------------|--|
| MLN_CLPBDFAIL   | Signals that a clipboard operation failed.   |
| MLN_MEMERROR    | Signals that the required storage cannot be obtained. The action that results in the increased storage requirement fails.  |
| MLN_SETFOCUS    | Sent whenever the MLE window receives the input focus.   |
| MLN_KILLFOCUS   | Sent whenever the MLE window loses the input focus.  |
| MLN_MARGIN      | <p>Whenever the user moves the mouse into the left, right top, or bottom margins, this message is sent to the owner of the window.</p> <p>If the owner returns an <i>rc</i> of TRUE, the mouse move is assumed to have been processed by the owner and no further action need be taken.</p> <p>If the owner returns an <i>rc</i> of FALSE, the MLE performs a default action appropriate to each different mouse action.</p> <p>The exceptions to this are all mouse messages that occur after a button-down inside the margin, until and including the matching button-up. Conceptually the drag (button-down until button-up) is a single macro event. Therefore, if FALSE is returned for a button-down event, no further margin notifications are given until after the drag has ended (button-up).</p> <p><b>Note:</b> If the application receives a notification of button-down in the margin and processes it, it must capture the mouse until the button-up event.</p> |
| MLN_SEARCHPAUSE | <p>This notification is sent periodically by the MLE, while an MLM_SEARCH message is being processed, to give an application the opportunity to stop excessively long searches, and to provide search progress information. The owner window can respond either with TRUE or FALSE. FALSE causes the MLE to continue searching; TRUE causes the MLE to stop the search immediately. For further information, see MLM_SEARCH</p>  |

## param2

### ulOver (ULONG)

Number of bytes that do not fit.

*param2* contains *ulOver* for a *usnotifycode* of MLN\_TEXTOVERFLOW.

### **pixOver (PIX)**

Linear distance of overflow in pels.

*param2* contains *pixOver* for a *usnotifycode* of MLN\_PIXHORZOVERFLOW or MLN\_PIXVERTOVERFLOW.

### **pErrInfo (POVERFLOW)**

Overflow error information structure.

*param2* contains *pErrInfo* for a *usnotifycode* of MLN\_OVERFLOW.

The *afErrInd* field of the MLEOVERFLOW structure can take one or more of the following values:

|                 |   |
|-----------------|---|
| MLFEFR_RESIZE   | The window is resized, and the format rectangle is tied to the window size and limited either horizontally, vertically, or both. The implicit change of the format rectangle to the new size does not contain the text. The format rectangle is made static at the previous size, and the MLESFR_MATCHWINDOW style is turned off until set again by the application. This is done in response to a WM_SIZE message, and therefore the multi-line entry field does not forward the return value from this notification message.  |
| MLFEFR_TABSTOP  | A tab stop location change is requested, and the text is limited either horizontally, vertically, or both. Changing the tab stops causes the text to exceed the limit. The tab stop change is rejected.   |
| MLFEFR_FONT     | A font change is requested, and the text is limited either horizontally, vertically, or both. Changing the font causes the text to exceed the limit. The font change is rejected.   |
| MLFEFR_WORDWRAP | The word-wrap state is requested to be changed, and the text is limited either horizontally, vertically, or both. Wrapping the text differently exceeds the limit, and the request is rejected. This happens in situations where the horizontal limit is not set, there are lines exceeding it, and word-wrap is being changed from off to on, such that it creates soft line breaks resulting in increased vertical size. This happens if word-wrap is being changed from on to off, and there is at least one line created by a soft line-break, such that when that line-break is removed, the full line (up to the hard line break) exceeds the horizontal limit. |
| MLFEFR_TEXT     | Text is changed by MLM_IMPORT, MLM_PASTE, MLM_CUT, MLM_UNDO, or MLM_DELETE, and the text is limited either horizontally, vertically, or both within the format rectangle. The change causes the text to exceed the format rectangle in a dimension that is limited. For   |

example, Delete and EOL joins text from two lines into one line long enough to exceed the horizontal limit.

**MLFETL\_TEXTBYTES** Text is changed by MLM\_IMPORT MLM\_PASTE, or MLM\_UNDO, and the text is limited to a maximum number of bytes. The change causes the text to exceed that maximum.

**ulErrInd (ULONG)**

Clipboard fail flag.

*param2* contains *ulErrInd* for a *usnotifycode* of MLN\_CLPBDFAIL.

**MLFCPBD\_TOOMUCHTEXT** Text amount exceeds clipboard capacity

**MLFCPBD\_CLPBDERROR** A clipboard error occurred.

**pmrg (PMARGSTRUCT)**

Margin structure.

*param2* contains *pmrg* for a *usnotifycode* of MLN\_MARGIN.

The left and right margins are defined as going all the way to the top and bottom such that the top and bottom margins are contained between them. Therefore, the corners are included in the sides.

*usMouMsg* contains the mouse message that signals the event.

*iptNear* contains the insertion point of the nearest point in the text. For situations where the nearest location is beyond the end of a line, the insertion point for the end of the line is returned. (The EOL character is considered to be beyond the end of the line.)

**iptSearchedTo (IPT)**

Current insertion point of search.

*param2* contains *iptSearchedTo* for a *usnotifycode* of MLN\_SEARCHPAUSE.

**ulReserved (ULONG)**

Reserved value, should be 0.

*param2* contains *ulReserved* for a *usnotifycode* of MLN\_HSCROLL, MLN\_VSCROLL, MLN\_CHANGE, MLN\_UNDOOVERFLOW, MLN\_MEMERROR, MLN\_SETFOCUS, or MLN\_KILLFOCUS.

## Returns

### ReturnCode

**rc** (BOOL)

Action taken by application.

*ReturnCode* contains *rc* for a *usnotifycode* of MLN\_TEXTOVERFLOW, MLN\_PIXHORZOVERFLOW, MLN\_PIXVERTOVERFLOW, MLN\_MARGIN, or MLN\_SEARCHPAUSE.

**TRUE** The multiline entry field control assumes that appropriate action has been taken by the application. Appropriate action depends on the MLN\_\* notification code, and is documented under the *usnotifycode* field.

**FALSE** The multiline entry field control assumes that the application has ignored this WM\_CONTROL (in Multiline Entry Fields) message, and takes action appropriate to the MLN\_\* notification code, as documented under the *usnotifycode* field.

**ulReserved** (ULONG)

Reserved value, should be 0.

*ReturnCode* contains *ulReserved* for a *usnotifycode* of MLN\_OVERFLOW, MLN\_HSCROLL, MLN\_VSCROLL, MLN\_CHANGE, MLN\_UNDOOVERFLOW, MLN\_CLPBDFAIL, MLN\_MEMERROR, MLN\_SETFOCUS, or MLN\_KILLFOCUS.

## Remarks

The multiline entry field control window procedure generates this message and sends it to its owner, informing the owner of the event.

*param2* depends on the MLN\_\* notification code.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

## Related Messages

- WM\_CONTROL

---

## Multi-Line Entry Field Window Messages

This section describes the multi-line entry field control window procedure actions on receiving the following messages.

---

### MLM\_CHARFROMLINE

This message returns the first insertion point on a given line.

#### Parameters

**param1**

**lLineNum** (LONG)

Line number of interest.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**iptFirst** (IPT)

First insertion point on line.

#### Remarks

For any line number, the insertion point just before the first character on that line is returned. If the line number is -1, the line containing the cursor is used.

The term line means a line on the display after the application of word-wrap. It does not mean a line as defined by the CR LF line-break sequence.

#### Default Processing

The default window procedure takes no action on this message, other than to set *iptFirst* to 0.

---

### MLM\_CLEAR

This message clears the current selection.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

## param2

### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### ulClear (ULONG)

Number of bytes deleted, counted in CF\_TEXT format.

## Remarks

The multi-line entry field control window procedure responds to this message by clearing the current selection and returning the number of bytes cleared.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulClear* to 0.

---

## MLM\_COPY

This message copies the current selection to the clipboard.

## Parameters

### param1

#### ulReserved (ULONG)

Reserved value, should be 0.

### param2

#### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### ulCopy (ULONG)

Number of bytes transferred, counted in CF\_TEXT format.

## Remarks

The multi-line entry field control window procedure responds to this message by copying the selected text to the clipboard. The text is translated to standard clipboard format, which is the same as exporting with MLE\_CFTTEXT format.

The text is placed on the clipboard as a single contiguous data segment. This restricts the amount to the maximum segment size (64KB).

This may cause an overflow, see MLN\_OVERFLOW.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulCopy* to 0.

---

## **MLM\_CUT**

This message copies the text that forms the current selection to the clipboard and then deletes it from the MLE control.

### **Parameters**

#### **param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### **param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Returns**

**ulCopy** (ULONG)

Number of bytes transferred, counted in CF\_TEXT format.

### **Remarks**

The multi-line entry field control window procedure responds to this message by copying the selected text to the clipboard and then deleting it. The text is translated to standard clipboard format, which is the same as exporting with MLE\_CFTEXT format.

The text is placed on the clipboard as a single contiguous data segment. This restricts the amount to the maximum segment size (64KB).

This may cause an overflow, see MLN\_OVERFLOW.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *ulCopy* to 0.



---

## MLM\_DELETE

This message deletes text.

### Parameters

**param1**

**iptBegin** (IPT)

Starting point of deletion.

**param2**

**ulDel** (ULONG)

Number of bytes to delete.

### Returns

**ulSuccess** (ULONG)

Number of bytes successfully deleted.

### Remarks

This message takes an insertion point and a length, and deletes that number of characters from the text. If the insertion point is -1, the selection is used and the effect is identical to the MLM\_CLEAR message.

This may cause an overflow, see MLN\_OVERFLOW.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulSuccess* to 0.

---

## MLM\_DISABLEREFRESH

This message disables screen refresh.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

## Remarks

This message disables screen refreshes. This allows an application to make changes throughout a document while avoiding unnecessary overhead caused by attempts to keep the screen display current. When an MLM\_ENABLELREFRESH message is sent, the screen display is brought up to date with the contents of the text.

While refresh is disabled, mouse and keyboard messages are processed by beeping and ignoring them, except for mouse moves, which do not beep; the mouse pointer changes to the system standard wait symbol (a clock face).

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_ENABLELREFRESH

This message enables screen refresh.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

0 Reserved value, 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE An error occurred.

## Remarks

This message enables screen refreshes. This allows an application to make changes throughout a document while avoiding unnecessary overhead caused by attempts to keep

the screen display current. When an MLM\_ENABLEREFRESH message is sent, the screen display is brought up to date with the contents of the text.

While refresh is disabled, mouse and keyboard messages are processed by beeping and ignoring them, except for mouse moves, which do not beep; the mouse pointer changes to the system standard wait symbol (a clock face).

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## **MLM\_EXPORT**

This message exports text to a buffer.

### **Parameters**

#### **param1**

##### **pBegin** (PIPT)

Starting point.

Updated to follow the last character exported.

#### **param2**

##### **pCopy** (PULONG)

Number of bytes being exported.

Decrement by the number of bytes actually exported.

### **Returns**

#### **ulSuccess** (ULONG)

Number of bytes successfully exported.

### **Remarks**

This message takes an insertion point and length as parameters, and copies text, starting from that insertion point, into the buffer set by MLM\_SETIMPORTEXP. Text is in the format set by MLM\_FORMAT. If the insertion point is -1, the selection is used for both *pBegin* and *pCopy*.

On return, *pBegin* is updated to follow the last byte exported, and the number of bytes to be exported is decremented by the number actually exported. This is done to prepare those parameter values for the next export. The return value indicates the number of bytes actually put into the buffer. This number is less than, or equal to, the buffer size (see MLM\_SETIMPORTEXP).

**Note:** All exports are done in full characters. Therefore, if either the length of the buffer or the number of bytes to be exported result in the last byte transferred being only half of a DBCS character, the MLE will *not* transfer that byte.

It returns the number of bytes placed in the export buffer.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulSuccess* to 0.

---

## MLM\_FORMAT

This message sets the format to be used for buffer importing and exporting.

### Parameters

**param1**

**usFormat** (USHORT)

Format to be used for import and export.

**MLFIE\_CFTEXT** Text format. Each line ends with a carriage-return/line-feed combination. Tab characters separate fields within a line. A NULL character signals the end of the data.

**MLFIE\_NOTRANS** Uses LF for line delineation, and guarantees that any text imported into the MLE in this format can be recovered in exactly the same form on export.

**MLFIE\_WINFMT** (Windows MLE format.) On import, recognizes CR LF as denoting hard line-breaks, and ignores the sequence CR CR LF. On export, uses CR LF to denote a hard line-break and CR CR LF to denote a soft line-break caused by word-wrapping.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usFormat** (USHORT)

Previous format value.

### Remarks

The default format is **MLFIE\_CFTEXT**.

The keyword **MLFIE\_RTF** is reserved.

## Default Processing

The default window procedure takes no action on this message, other than to set *usFormat* to 0.

---

## MLM\_IMPORT

This message imports text from a buffer.

### Parameters

**param1**

**pBegin** (PIPT)

Insertion point.

Updated to insertion point following last insert.

**param2**

**ulCopy** (ULONG)

Number of bytes in buffer.

### Returns

**ulSuccess** (ULONG)

Number of bytes successfully inserted.

### Remarks

This message takes an insertion point and length as parameters. It assumes a buffer has been set using `MLM_SETIMPORTEXPORT`, and inserts the contents of the buffer at the insertion point in the text. The contents are interpreted as being in the format set by `MLM_FORMAT`. If the insertion point is `-1`, the cursor point is used.

The insertion point *pBegin* is updated by the MLE to the point after the last character imported. This provides the application with the location for the next import.

The return value indicates how many bytes were actually transferred.

All imports are done in full characters, therefore, if the number of bytes to be imported results in the last byte transferred being only half of a DBCS character, or part of a line-break sequence (CR LF or CR CR LF), the MLE does not transfer that byte. If the return value indicates that less than the full amount was transferred, a check must be made to determine if it is the beginning of a multi-byte sequence, and if so, the parts must be mated and imported as a whole.

This can cause an overflow, see `MLN_OVERFLOW`.

**Note:** The buffer is not zero-terminated; NULL characters can be inserted into the text.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulSuccess* to 0.

---

## MLM\_INSERT

This message deletes the current selection and replaces it with a text string.

### Parameters

**param1**

**pchText** (PCHAR)

Null-terminated text string.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulCount** (ULONG)

Number of bytes actually inserted.

### Remarks

This message inserts the text string at the current selection, deleting that selection in the same manner as typing at the keyboard would. The text string must be in CF\_TEXT format (or one of the formats acceptable to MLM\_IMPORT) and null-terminated. The line-break (CR LF, LF, and so on) is counted as one byte, regardless of the number of bytes occupied in the buffer, and the null terminator is not counted.

This interacts with the format rectangle and text limits, and a return of less than the full count can be the result. If so, a notification message is sent.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulCount* to 0.

---

## MLM\_LINEFROMCHAR

This message returns the line number corresponding to a given insertion point.

### Parameters

**param1**

**iptFirst (IPT)**

Insertion point of interest.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### Returns

**ILineNum (LONG)**

Line number of insertion point.

### Remarks

For any insertion point, the corresponding line number is returned. If the insertion point is -1, the number of the line containing the first insertion point of the selection is returned.

The term line means a line on the display after the application of word-wrap. It does not mean a line as defined by the CR LF line-break sequence.

### Default Processing

The default window procedure takes no action on this message, other than to set *ILineNum* to 0.

---

## MLM\_PASTE

This message replaces the text that forms the current selection, with text from the clipboard.

### Parameters

**param1**

**ulReserved (ULONG)**

Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

### **ulCopy** (ULONG)

Number of bytes transferred, counted in. CF\_TEXT format.

## Remarks

The multi-line entry field control window procedure responds to this message by replacing the selected text with text from the clipboard. The text is translated from standard clipboard format, which is the same as importing with MLE\_CFTTEXT format.

The text is assumed to be in the clipboard as a single contiguous data segment. This restricts the amount to the maximum segment size (64Kb).

This can cause an overflow, see MLN\_OVERFLOW.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulCopy* to 0.

---

## MLM\_QUERYBACKCOLOR

This message queries the background color.

## Parameters

### **param1**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### **param2**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **IColor** (LONG)

Text color.

## Remarks

This message returns the color in which the background is to be drawn.

The color values are the same as those used by GpiSetColor.

## Default Processing

The default window procedure takes no action on this message, other than to set *IColor* to 0.



---

## MLM\_QUERYCHANGED

This message queries the changed flag.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Current changed status.

**TRUE** Text has changed since the last time that the change flag was cleared.

**FALSE** Text has not changed since the last time that the change flag was cleared.

### Remarks

The multi-line entry field control window procedure responds to this message by returning the changed flag for the text without altering it. See also MLN\_CHANGE.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to 0 (FALSE).

---

## MLM\_QUERYFIRSTCHAR

This message queries the first visible character.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**iptFVC** (IPT)

First visible character.

## Remarks

Returns the insertion point immediately preceding the character visible in the upper left-hand corner of the screen. If a partial character is displayed, that character counts as the first visible character.

**Note:** In situations where no character is visible, because the text is scrolled to the right beyond the end of the top line, this returns the insertion point of the last character on the line (EOL not considered). In situations where there are no characters on the line, the insertion point at the beginning is returned.

## Default Processing

The default window procedure takes no action on this message, other than to set *iptFVC* to 0.

---

## MLM\_QUERYFONT

This message queries which font is in use.

## Parameters

**param1**

**pFattrs** (PFATTRS)

Font attribute structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

System font indicator.

TRUE The system font is in use.

FALSE The system font is not in use.

## Remarks

This message puts the attributes of the current drawing font into the font attribute structure.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_QUERYFORMATLINELENGTH

This message returns the number of bytes to end of line after formatting has been applied.

### Parameters

**param1**

**iptStart** (IPT)

Insertion point to count from.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**iptLine** (IPT)

Count of bytes to end of line.

### Remarks

For any insertion point, the number of bytes between that insertion point and the end of the line is returned, after the current formatting is applied. If the insertion point is -1, the cursor position is used. This message differs from MLM\_QUERYLINELENGTH in that the byte count returned reflects the effects of the current formatting set by MLM\_FORMAT.

### Default Processing

The default window procedure takes no action on this message, other than to set *iptLine* to 0.

---

## MLM\_QUERYFORMATRECT

This message queries the format dimensions and mode.

### Parameters

**param1**

**pFormatRect** (PPOINTL)

Format dimensions.

The size of the current limiting dimensions.

## param2

### **flFlags** (ULONG)

Flags governing interpretation of dimensions.

An array of MLFFMTRECT\_\* flags defined under the *flFlags* field of the MLM\_SETFORMATRECT message.

## **Returns**

### **ulReserved** (ULONG)

Reserved value.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

---

## **MLM\_QUERYFORMATTEXTLENGTH**

This message returns the length of a specified range of characters after the current formatting has been applied.

## **Parameters**

### **param1**

#### **iptStart** (IPT)

Insertion point to start from.

### **param2**

#### **ulScan** (ULONG)

Number of characters to convert to bytes.

|            |   |
|------------|---|
| 0xFFFFFFFF | Convert until end of line               |
| other      | Convert specified number of characters. |

## **Returns**

### **ulText** (ULONG)

Count of bytes in text after formatting.

## Remarks

This message returns the length in bytes of a range of characters after the current formatting is applied. This differs from `MLM_QUERYTEXTLENGTH` in that:

- A range of insertion points can be queried.
- The byte count returned reflects the effects of the current formatting set by `MLM_FORMAT`.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulText* to 0.

---

## MLM\_QUERYIMPORTEXP

This message queries the current transfer buffer.

### Parameters

**param1**

**Buff** (PVOID \*)  
Transfer buffer.

**param2**

**puLength** (PULONG)  
Size of transfer buffer in bytes.

### Returns

**rc** (ULONG)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

### Remarks

This message returns the values from the most recent `MLM_SETIMPORTEXP`, or 0 for either value if it has not been set.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to 0 (FALSE).

---

## MLM\_QUERYLINECOUNT

This message queries the number of lines of text.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulLines** (ULONG)

The number of lines of text.

### Remarks

The term line means a line on the display after the application of word-wrap. It does not mean a line as defined by the CR LF line-break sequence.

The multi-line edit control always maintains one CR LF line-break in the buffer, therefore the number of lines returned may be one greater than the number actually visible.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulLines* to 0.

---

## MLM\_QUERYLINELENGTH

This message returns the number of bytes between a given insertion point and the end of line.

### Parameters

**param1**

**iptStart** (IPT)

Insertion point to count from.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

**iptLine (IPT)**

Count of bytes to end of line.

### **Remarks**

For any insertion point, the number of bytes between that insertion point and the end of the line is returned. If the insertion point is -1, the cursor position is used. If the line contains a hard line-break, it is counted as one byte.

The term line means a line on the display after the application of word-wrap. It does not mean a line as defined by the CR LF line-break sequence.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *iptLine* to 0.

---

## **MLM\_QUERYREADONLY**

This message queries the read-only mode.

### **Parameters**

**param1**

**ulReserved (ULONG)**

Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

**rc (BOOL)**

Current read-only status.

TRUE Read-only mode is set.

FALSE Read-only mode is cleared.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_QUERYSEL

This message returns the location of the selection.

### Parameters

#### param1

##### **usQueryMode** (USHORT)

Query Mode.

|                 |  |
|-----------------|--|
| MLFQS_MINMAXSEL | Return both minimum and maximum points of selection in a format compatible with the EM_QUERYSEL message. |
| MLFQS_MINSEL    | Return minimum insertion point of selection.   |
| MLFQS_MAXSEL    | Return maximum insertion point of selection.   |
| MLFQS_ANCHORSEL | Return anchor point of selection.  |
| MLFQS_CURSORSEL | Return cursor point of selection.  |

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### ReturnCode

##### **sMinSel** (SHORT)

Minimum insertion point of selection.

This value is rounded down to 65 535, if necessary.

*ReturnCode* contains *sMinSel* and *sMaxSel* for a *usQueryMode* of MLFQS\_MINMAXSEL.

##### **sMaxSel** (SHORT)

Maximum insertion point of selection.

This value is rounded down to 65 535 if necessary.

*ReturnCode* contains *sMinSel* and *sMaxSel* for a *usQueryMode* of MLFQS\_MINMAXSEL.



## ipt (IPT)

Requested insertion point.

*ReturnCode* contains *ipt* for a *usQueryMode* of MLFQS\_MINSEL, MLFQS\_MAXSEL, MLFQS\_ANCHORSEL, or MLFQS\_CURSORSEL.

## Remarks

This message returns the location of the selection in several different forms. The insertion points lie between characters, and start at a zero origin before the first character in the MLE. Subtracting the minimum from the maximum gives the number of characters in the selection. *This is not necessarily the number of bytes of ASCII.* The line-break character is a CR LF (2 bytes) and all DBCS characters are 2 bytes. To determine the number of bytes, use MLM\_QUERYFORMATTEXTLENGTH, being sure that the format choice set by MLM\_FORMAT is set to what is used when the data is exported from the MLE (for example, MLE\_CFTTEXT for MLM\_QUERYSELTEXT).

Note the following:

- If anchor point > cursor point, minimum point = cursor point and maximum point = anchor point.
- If anchor point < cursor point, minimum point = anchor point and maximum point = cursor point.

## Default Processing

The default window procedure takes no action on this message, other than to set *ReturnCode* to 0.

## Examples

This example sends two MLM\_QUERYSEL messages to obtain the beginning and ending points of the current selection, sends an MLM\_SETIMPORTEXPORT message to set up the export buffer, and then sends an MLM\_EXPORT message to export the selection into the buffer.

```
LONG lStart, cch;  
CHAR szBuf[500];  
  
lStart = (LONG) WinSendMessage(hwndMle, MLM_QUERYSEL,  
    (LPARAM) MLFQS_MINSEL, (LPARAM) 0L);  
cch = lStart - (LONG) WinSendMessage(hwndMle, MLM_QUERYSEL,  
    (LPARAM) MLFQS_MAXSEL, (LPARAM) 0L);  
WinSendMessage(hwndMle, MLM_SETIMPORTEXPORT,  
    (LPARAM) szBuf, (LPARAM) sizeof(szBuf));  
WinSendMessage(hwndMle, MLM_EXPORT, (LPARAM) &lStart, (LPARAM) &cch);
```

---

## MLM\_QUERYSELTEXT

This message copies the currently selected text into a buffer.

### Parameters

**param1**

**pchBuff** (PCHAR)

Character buffer for text string.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulCount** (ULONG)

Number of bytes to put into text string.

### Remarks

This message copies the currently selected text into the buffer pointed to by *pchBuff*. The text string is null-terminated. The byte count includes the text in CF\_TEXT format (CR LF) and the null terminator.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulCount* to 0.

---

## MLM\_QUERYTABSTOP

This message queries the pel interval at which tab stops are placed.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **pixTabset** (PIX)

Tab width in pels.

< 0 An error occurred.

Other The pel interval at which tab stops are placed.

## Remarks

This message fails and returns a negative value, if the reserved values are not 0.

## Default Processing

The default window procedure takes no action on this message, other than to set *pixTabset* to 0.

---

## MLM\_QUERYTEXTCOLOR

This message queries the text color.

## Parameters

### **param1**

**uiReserved** (ULONG)

Reserved value, should be 0.

### **param2**

**uiReserved** (ULONG)

Reserved value, should be 0.

## Returns

### **IColor** (LONG)

Text color.

## Remarks

This message returns the color in which text is to be drawn.

The color values are the same as those used by *GpiSetColor*.

## Default Processing

The default window procedure takes no action on this message, other than to set *IColor* to 0.

---

## MLM\_QUERYTEXTLENGTH

This message returns the number of characters in the text.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**iptText** (IPT)

Count of text in bytes.

### Remarks

This message returns the number of characters in the text. Hard line-breaks are counted as 1 and soft line-breaks as 0.

This message differs from the `WinQueryWindowTextLength` call in that it returns a LONG.

### Default Processing

The default window procedure takes no action on this message, other than to set *iptText* to 0.

---

## MLM\_QUERYTEXTLIMIT

This message queries the maximum number of bytes that a multi-line entry field control can contain.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ISize** (LONG)

Maximum number of bytes allowed in the MLE.

## Remarks

The multi-line entry field control window procedure responds to this message by returning the current limit set, either by default, or by `MLM_SETTEXTLIMIT`. If the limit is unbounded, a non-positive value is returned.

## Default Processing

The default window procedure takes no action on this message, other than to set *ISize* to 0.

---

## MLM\_QUERYUNDO

This message queries the undo or redo operations that are possible.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ReturnCode**

**usOperation** (USHORT)

Operation that can be undone or redone.

|                  |  |
|------------------|--|
| 0                | An undo or redo operation is not possible.   |
| WM_CHAR          | A WM_CHAR message, or messages for a simple string of keystrokes, can be undone or redone.   |
| MLM_SETFONT      | A MLM_SETFONT message can be undone or redone.   |
| MLM_SETTEXTCOLOR | A MLM_SETTEXTCOLOR message can be undone or redone for both background and foreground color. |
| MLM_CUT          | A MLM_CUT message can be undone or redone.   |
| MLM_PASTE        | A MLM_PASTE message can be undone or redone.   |
| MLM_CLEAR        | A MLM_CLEAR message can be undone or redone.   |

**rc** (BOOL)

Undo or redo indicator.

TRUE    An undo is possible.

FALSE   A redo is possible.

### Default Processing

The default window procedure takes no action on this message, other than to set *reply* to 0.

---

## MLM\_QUERYWRAP

This message queries the wrap flag.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Wrap flag.

TRUE    Word-wrap enabled

FALSE   Word-wrap disabled.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_RESETUNDO

This message resets the undo state to indicate that no undo operations are possible.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

## param2

### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### ReturnCode

### usOperation (USHORT)

Operation that can be undone or redone.

|                  |  |
|------------------|--|
| 0                | An undo or redo operation is not possible.   |
| WM_CHAR          | A WM_CHAR message, or messages for a simple string of keystrokes, can be undone or redone.   |
| MLM_SETFONT      | A MLM_SETFONT message can be undone or redone.   |
| MLM_SETTEXTCOLOR | A MLM_SETTEXTCOLOR message can be undone or redone for both background and foreground color. |
| MLM_CUT          | A MLM_CUT message can be undone or redone.   |
| MLM_PASTE        | A MLM_PASTE message can be undone or redone.   |
| MLM_CLEAR        | A MLM_CLEAR message can be undone or redone.   |

### rc (BOOL)

Undo or redo indicator.

|       |                      |
|-------|----------------------|
| TRUE  | An undo is possible. |
| FALSE | A redo is possible.  |

## Remarks

This message resets the undo state of the MLE to indicate that the last operation cannot be undone (null return from MLM\_QUERYUNDO). This can be used by the application when it performs an operation that it can undo, that supersedes the last MLE operation. The application can then reset its own undo state upon receipt of an MLN\_CHANGE, indicating that later changes have occurred through the MLE.

## Default Processing

The default window procedure takes no action on this message, other than to set *ReturnCode* to 0.

---

## MLM\_SEARCH

This message searches for a specified text string.

### Parameters

#### param1

##### ulStyle (ULONG)

Style flags.

- |                         |  |
|-------------------------|--|
| MLFSEARCH_CASESENSITIVE | If set, only exact matches are considered a successful match. If not set, any case-combination of the correct characters in the correct sequence is considered a successful match.   |
| MLFSEARCH_SELECTMATCH   | If set, the MLE selects the text and scrolls it into view when found, just as if the application had sent an MLM_SETSEL message. This is not done if MLFSEARCH_CHANGEALL is also indicated.  |
| MLFSEARCH_CHANGEALL     | Using the MLE_SEARCHDATA structure specified in <i>pse</i> , all occurrences of <i>pchFind</i> are found, searching from <i>iptStart</i> to <i>iptStop</i> , and replacing them with <i>pchReplace</i> . If this style is selected, the <i>cchFound</i> field has no meaning, and the <i>iptStart</i> value points to the place where the search stopped, or is the same as <i>iptStop</i> because the search has not been stopped at any of the found strings. The current cursor location is not moved. However, any existing selection is deselected. |

#### param2

##### pse (PMLE\_SEARCHDATA)

Search specification structure.

### Returns

#### rc (BOOL)

Success indicator.

- |       |                              |
|-------|------------------------------|
| TRUE  | The search was successful.   |
| FALSE | The search was unsuccessful. |



## Remarks

This message searches the MLE text for a specified string, starting at a specified insertion point and continuing until the second specified insertion point has been reached, or the requested string has been matched.

When an MLM\_SEARCH message is sent, the text is scanned starting with the character that follows the insertion point indicated in the *iptStart* field of the MLM\_SEARCHDATA structure. The search proceeds until the point indicated in the *iptStop* field, until a match is found, or until TRUE is returned from MLN\_SEARCHPAUSE notification (see WM\_CONTROL (in Multiline Entry Fields)). If a negative value is specified for the *iptStart*, the current cursor point is used. If a negative value is specified for *iptStop*, the end of the text is used. If *iptStop*, is less than or equal to *iptStart*, after performing the two indicated substitutions, the search wraps from the end of the text to the beginning of the text.

If the MLFSEARCH\_CASESENSITIVE option is specified, the bytes of the search string must exactly match those in the text. If MLFSEARCH\_CASESENSITIVE is not specified, the WinUpperChar of the search string must match the WinUpperChar of the text.

When a match is found, the *iptStart* field of the search specification structure is set to indicate the insertion point immediately preceding the first character of the match, and the *cchFind* field is set to indicate the number of characters in the match. The cursor selection is not altered unless MLFSEARCH\_SELECTMATCH is specified. If it is, an MLM\_SETSEL is done with the anchor point at *iptStart* and the cursor at *iptStart + cchFind*.

While searching, the MLE occasionally sends an MLN\_SEARCHPAUSE notification message. If the owner responds to this message with the value TRUE, the MLE stops the search. When a search is stopped from MLN\_SEARCHPAUSE, *iptStart* is set to the point where the search terminated. If the response is FALSE, the search continues (see also the definition of MLN\_SEARCHPAUSE). The interval at which MLN\_SEARCHPAUSE notifications are sent is implementation-dependent, but must not exceed reasonable user-response thresholds, nor should it be so often as to introduce undue messaging overhead. Sending this notification every half second is a reasonable compromise.

When no match is found the *iptStart* value is unchanged.

If the application needs to continue the search, the proper way is to change the *iptStart* value to be the point following the string found, adjusting for any text changes done after the search that may have moved the relative location of the point.

Applications using this message are advised to change the system pointer to the wait icon (clock face) if it is expected that the search will take some time.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Examples

This example searches for all occurrences of the word "Bonnie" and replaces them with the word "Jeannette."

```
MLE_SEARCHDATA search;
search.cb = sizeof(search);
search.pchFind = "bonnie";
search.pchReplace = "jeannette";
search.cchFind = 6;
search.cchReplace = 9;
search.iptStart = 0; /* from the beginning of the text */
search.iptStop = -1; /* to the end of the text */
WinSendMessage(hwndMle, MLM_SEARCH, MLFSEARCH_CHANGEALL, (MPARAM) &search)
```

---

## MLM\_SETBACKCOLOR

This message sets the background color.

### Parameters

**param1**

**IColor** (LONG)  
Color.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Returns

**IOldColor** (LONG)  
Color previously used.

### Remarks

This message sets the color in which the MLE background is to be drawn, and updates the display as necessary.

The color values are the same as those used by `GpiSetColor`.

### Default Processing

The default window procedure takes no action on this message, other than to set *IOldColor* to 0.

---

## MLM\_SETCHANGED

This message sets or clears the changed flag.

### Parameters

**param1**

**usChangedNew** (USHORT)

Value to set changed flag to.

TRUE Changed flag set.

FALSE Changed flag cleared.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Changed status before message was processed.

TRUE Text has changed since the last time that the change flag was cleared.

FALSE Text has not changed since the last time that the change flag was cleared.

### Remarks

This message can generate a MLN\_CHANGE notification.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_SETFIRSTCHAR

This message sets the first visible character.

### Parameters

**param1**

**iptFVC** (IPT)

Insertion point to place in top left-hand corner.

## param2

### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### rc (BOOL)

Success indicator.

TRUE Successful completion

FALSE An error occurred.

## Remarks

This message scrolls the text to place the character following the insertion point into the upper left-hand corner of the window. If the insertion point specified is beyond the end of a line, or the end of the file, it is resolved in the same way as it is for a mouse click.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_SETFONT

This message sets a font.

## Parameters

### param1

#### pFattrs (PFATTRS)

Font attribute structure.

NULL The system font is set.

other The specified font is set.

### param2

#### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### rc (BOOL)

Success indicator.

TRUE The font was successfully set.

FALSE An error occurred.

## Remarks

For any *PFATTRS*, this message sets the display to use the appropriate font. If NULL, the system font is used. The screen is updated appropriately.

This can cause an overflow, see `MLN_OVERFLOW`.

When setting an outline font it is necessary to ensure that the *FATTRS* structure contains the correct maximum baseline extent and average character width for the desired point size and that the font use is marked as `FATTR_FONTUSE_TRANSFORMABLE`.

Baseline extent and character width are calculated by multiplying the desired point size by the current display device font resolution (`CAPS_VERTICAL_FONT_RES` and `CAPS_HORIZONTAL_FONT_RES`; see `DevQueryCaps`) and dividing by 72, the number of points in an inch.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to `FALSE`.

## Examples

This example retrieves the current font information, changes it to italic, and sets it using the `MLM_SETFONT` message.

```
FATTRS fat;
fat.usRecordLength = sizeof(FATTRS);
WinSendMsg(hwndMle, MLM_QUERYFONT, (MPARAM) &fat, (MPARAM) 0L);
fat.fsSelection = FATTR_SEL_ITALIC;
WinSendMsg(hwndMle, MLM_SETFONT, (MPARAM) &fat, (MPARAM) 0);
```

---

## MLM\_SETFORMATRECT

This message sets the format dimensions and mode.

### Parameters

**param1**

**pFormatRect** (PPOINTL)

New format dimensions.

**NULL** A null value sets both dimensions to the current window size.

**other** The structure is a pair of LONGs designating the diagonally-opposite corner of the rectangle, assuming 0,0 for the first. Therefore, they are the width and height in pels of the format rectangle. These dimensions are used as the word-wrap and text-size limiting boundaries. Negative values for either

dimension cause the MLE to substitute the current window size (the MLE window rectangle minus margins).

If the rectangle specified has either, or both, of the limits set, and the size is inadequate to contain the text, *rc* is set to FALSE and the rectangle dimensions are replaced with the overflow amounts.

## param2

### fFlags (ULONG)

Flags governing interpretation of dimensions.

|                        |  |
|------------------------|--|
| MLFFMTRECT_MATCHWINDOW | The dimensions of the format rectangle are always to be kept the same as the window size minus the margins. This causes the MLE implicitly to do a MLM_SETFORMATRECT each time the window is resized, and effectively causes any other dimensions to be ignored. Resizing of the window can cause this setting to be automatically negated (see MLN_OVERFLOW). |
| MLFFMTRECT_LIMITHORZ   | The width of any line in the MLE cannot exceed the given horizontal dimension. If word-wrap is on, this limit has no effect. Word-wrap can result in trailing blanks beyond the right limit. These do not cause an overflow notification.  |
| MLFFMTRECT_LIMITVERT   | The vertical height of the total text, as displayed, is limited to that which fits totally within the vertical dimension of the format rectangle.  |

## Returns

*rc* (BOOL)

Success indicator.

|       |                       |
|-------|-----------------------|
| TRUE  | Successful completion |
| FALSE | An error occurred.    |

## Remarks

The multi-line entry field control window procedure responds to this message by setting formatting dimensions and mode.

Any addition of text that causes the text to exceed the rectangle limits causes a notification before proceeding (see MLN\_PIXHORZOVERFLOW and MLN\_PIXVERTOVERFLOW).

Any activity that would cause the rectangle to be unable to contain the existing text (resize, undo, increasing font size, or word-wrap on or off) is rejected and results in a notification message for information (see `MLN_OVERFLOW`).

### **Default Processing**

The default window procedure takes no action on this message, other than to set `rc` to `FALSE`.

---

## MLM\_SETREADONLY

This message sets or clears read-only mode.

### Parameters

**param1**

**usReadOnly** (USHORT)

New read-only value.

TRUE Read-only mode set.

FALSE Read-only mode cleared.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Previous read-only value.

TRUE Read-only mode was set.

FALSE Read-only mode was cleared.

### Remarks

When read-only mode is set, characters typed at the keyboard do not get inserted into the MLE text. The API insertion interface, however, is still functional, as are selection-manipulation activities and copy-to-clipboard operations. This is useful as a means of preventing text modification (such as in a help system), and for providing a minimal blocking printing semaphore.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_SETIMPORTEXPOR

This message sets the current transfer buffer.

### Parameters

**param1**

**pBuff** (PCHAR)

Transfer buffer.



## **param2**

### **ulLength (ULONG)**

Size of transfer buffer in bytes.

## **Returns**

### **rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE An error occurred.

## **Remarks**

Given a far pointer to a buffer, and the size of the buffer, this message sets it as the current transfer buffer for the MLE. This buffer is used by the MLM\_IMPORT and MLM\_EXPORT messages. The system segment limit must be observed when specifying the buffer size.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

---

## MLM\_SETSEL

This message sets a selection.

### Parameters

**param1**

**iptAnchor** (IPT)

Insertion point for new anchor point.

**param2**

**iptCursor** (IPT)

Insertion point for new cursor point.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Selection successfully set

FALSE An error occurred.

### Remarks

This message sets the anchor and cursor points. The screen display is updated appropriately, ensuring that the cursor point is visible (which may involve scrolling). Note that the text cursor and inversion are not displayed if the MLE window does not have the input focus. A negative value for a point leaves that point alone.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Examples

This example highlights the second, third, and fourth characters of the text, and places the cursor to the right of the fourth character.

```
WinSendMessage(hwndMle, MLM_SETSEL, (LPARAM) 1L, (LPARAM) 4L);
```

---

## MLM\_SETTABSTOP

This message sets the pel interval at which tab stops are placed.

### Parameters

#### param1

**pixTab** (PIX)

Pel interval for tab stops.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**pixTabset** (PIX)

Success indicator.

< 0 An error occurred.

Other The value to which the width was set.

### Remarks

This message fails if the reserved value is not 0.

This message can cause an overflow, see MLN\_OVERFLOW.

### Default Processing

The default window procedure takes no action on this message, other than to set *pixTabset* to 0.

---

## MLM\_SETTEXTCOLOR

This message sets the text color.

### Parameters

#### param1

**IColor** (LONG)

Color.

## param2

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **IOldColor (LONG)**

Color previously used.

## **Remarks**

This message sets the color in which the MLE text is to be drawn, and updates the display as necessary.

The color values are the same as those used by `GpiSetColor`.

## **Default Processing**

The default window procedure takes no action on this message, other than to set *IOldColor* to 0.

---

## **MLM\_SETTEXTLIMIT**

This message sets the maximum number of bytes that a multi-line entry field control can contain.

## **Parameters**

### **param1**

#### **ISize (LONG)**

Maximum number of characters in `MLFIE_NOTRANS` format.

### **param2**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **ulFit (ULONG)**

Success indicator.

0 Successful completion. Current text fits within the new limit.

Other The number of bytes by which the current text exceeds the proposed limit. The limit is not changed.

## Remarks

The multi-line entry field control window procedure responds to this message by limiting the text size to *lSize* bytes. Text size is calculated using the `MLFIE_NOTRANS` format. Note that this is bytes and *not* characters; DBCS programmers should calculate accordingly.

This message returns 0 if the text limit exceeds or is equal to the existing text. Otherwise it returns the number of bytes by which the text would have overflowed, and does not change the limit.

The default, which is unbounded, can be specified by entering a non-positive limit.

## Default Processing

The default window procedure takes no action on this message, other than to set *ulFit* to 0.

---

## MLM\_SETWRAP

This message sets the wrap flag.

### Parameters

**param1**

**usWrap** (USHORT)

New value for wrap flag.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE An error occurred.

## Remarks

The multi-line entry field control window procedure responds to this message by setting the word wrap mode and updating the screen as appropriate.

When word-wrap is turned on, the text is wrapped to fit the formatting rectangle width. When word-wrap is turned off, the text is allowed to trail off to the right until it reaches an end-of-line marker.

Word-wrapping is defined as follows. Words are sequences of non-white-space characters (white-space characters are space, line break, and tab). When word-wrapping is enabled, the whole word must appear on one line within the formatting rectangle, unless the word by

itself is too long to fit. In this case the word is split following the last character that fits, and the remainder starts a new line.

This definition then applies recursively to the remainder of the word. The word continues to be visible. For editing purposes (for example, for word-selection) the word is viewed as a single word drawn over multiple lines.

Blank characters are always accumulated onto the current line, even if they exceed the horizontal formatting dimension, that is, blanks are allowed to trail off the right-hand edge. Line-break characters are also allowed to exceed the horizontal dimension, and any subsequent text must begin on a new line. The line-break following a line-break character is sometimes referred to as a hard line-break. Other line breaks, due to word-wrapping, and not to explicit formatting characters, are referred to as soft line-breaks.

Tab characters must always be visible. If a tab character occurs after the last tab stop within the horizontal formatting dimension, a soft line-break occurs after the tab.

This message can cause an overflow, see `MLN_OVERFLOW`.

### **Default Processing**

The default window procedure takes no action on this message, other than to set `rc` to `FALSE`.

---

## **MLM\_UNDO**

This message performs any available undo operation.

### **Parameters**

**param1**

**ulReserved (ULONG)**

Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

**rc (USHORT)**

Success indicator.

TRUE An undo operation was performed.

FALSE No undo operation was performed.

## Remarks

The last operation is undone (note that an undo can be undone).

This can cause an overflow, see `MLN_OVERFLOW`.

## Default Processing

The default window procedure takes no action on this message, other than to set `rc` to `FALSE`.

---

## WM\_BUTTON1DBLCLK (in Multiline Entry Fields)

For the cause of this message, see “WM\_BUTTON1DBLCLK” on page 10-12.

For a description of the parameters, see “WM\_BUTTON1DBLCLK” on page 10-12.

## Remarks

This message indicates that mouse button 1 has clicked twice within the system double-click time.

### Double-Click

If the click point is in the middle of a non-white-space character, the token (word) surrounding the clicked-on character, and any trailing spaces, are selected. If the click point is in a space character, the previous word (along with the trailing spaces including the clicked-on space) is selected. If there is no preceding word (either because the spaces are at the beginning of the text or immediately follow a line-break character) the run of spaces is selected. If the click point is on a tab or line-break character, that character is selected.

### Shift-Double-Click

Double-clicking while the Shift key is pressed leaves the anchor point alone, and moves the cursor point to the beginning or end of the clicked-on token. If the click point is before the anchor point in the text, the cursor point is moved to the beginning of the surrounding word, otherwise, the cursor point is moved to the end of the surrounding word. When shift-double-clicking, the selection is extended to include the token that was double-clicked on.

### Margin Mouse Event

All mouse events in a margin cause the MLE to send a `MLN_MARGIN` notification to the owner window of the MLE. This message has, as its parameters, the original mouse message. The owner can process the notification or not. If the owner does not process the message, the event is treated as if it occurred on the closest point in the text.

## Default Processing

The default window procedure takes no action on this message, other than to set `rc` to `FALSE`.

## Related Messages

- `WM_BUTTON1DBLCLK`

---

## **WM\_BUTTON1DOWN (in Multiline Entry Fields)**

For the cause of this message, see “WM\_BUTTON1DOWN” on page 10-13.

For a description of the parameters, see “WM\_BUTTON1DOWN” on page 10-13.

### **Remarks**

This message delimits mouse button click events. Between a button-down and a button-up event, the mouse is considered to be dragging. A mouse click is considered to happen on button-down, and dragging is terminated by a button-up.

#### **Click**

Clicking in the text sets the cursor and anchor points to the nearest insertion point. If the MLE is in overtype mode, the anchor is extended one character further in the text, subject to the end-of-text and new-line boundary conditions, defined under WM\_CHAR (in Multiline Entry Fields).

#### **Shift-Click**

Clicking while the shift key is held down sets the cursor point to the nearest insertion point, while leaving the anchor point alone.

#### **Margin Mouse Event**

All mouse events in a margin cause the MLE to send a MLN\_MARGIN notification to the owner window of the MLE. This message has, as its parameters, the original mouse message. The owner can process the notification or not. If the owner does not process the message, the event is treated as if it occurred on the closest point in the text.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_BUTTON1DOWN

---

## **WM\_BUTTON1UP (in Multiline Entry Fields)**

For the cause of this message, see “WM\_BUTTON1UP” on page 10-16.

For a description of the parameters, see “WM\_BUTTON1UP” on page 10-16.

### **Remarks**

This message delimits mouse button click events. Between a button-down and a button-up event the mouse is considered to be dragging. A mouse click is considered to happen on button-down, and dragging is terminated by a button-up.

#### **Margin Mouse Event**

All mouse events in a margin cause the MLE to send a MLN\_MARGIN notification to the owner window of the MLE. This message has, as its parameters, the original mouse message. The owner can process the notification or not. If the owner does not process the message, the event is treated as if it occurred on the closest point in the text.



## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_BUTTON1UP

---

## WM\_CHAR (in Multiline Entry Fields)

For the cause of this message, see "WM\_CHAR" on page 10-32.

For a description of the parameters, see "WM\_CHAR" on page 10-32.

### Remarks

The behavior of the MLE, when typing, depends on whether it is in insert or overtype mode, and whether the selection is empty or not. The selection is defined to be empty when the cursor point is equal to the anchor point.

When a character is typed, it replaces the current selection. If the selection is empty, the character is viewed as replacing nothing, so the character is effectively inserted into the text. If one or more characters are selected, those characters are deleted from the text and replaced by the typed character.

If the MLE is in insert mode, the cursor and anchor points are moved to immediately follow the newly typed character.

If the MLE is in overtype mode, the cursor is moved to immediately follow the newly typed character. If there is no character after the cursor (the new character is at the end of the text) or if the character after the cursor is a line-break character, the anchor is set to be equal to the cursor point. In any other case, the anchor is extended one character past the cursor point, defining the next character as the current selection.

If the typing causes the cursor to go off the screen in any direction, the display is automatically scrolled. If word-wrap is on, text continues on a new line, otherwise, the screen is scrolled horizontally.

Scrolling of the text in the window is independent of cursor movement. The cursor and selection remain unaltered at the same location within the text during all scrolling but the converse is not true. Any movement of the cursor causes auto-scrolling, if necessary, to ensure that the text location of the cursor is visible within the window.

**Tabs:** Tabs are represented as a single character in the text model, and are displayed as enough white-space to reach the next tab stop. Tab stops are set at *pel* intervals, starting with zero and occurring every *n* pels, where *n* is a value set by the MLM\_SETTABSTOP message, and defaulting to eight times the average character width of the system font.

When a tab is drawn, it uses the number of pels defined by the following formula:

$$pelWidth = pelTab - (pelDraw \text{ mod } pelTab)$$

where `pelTab` is the tab interval, in pels, and `pelDraw` is the pel at which drawing is to begin.

**Return:** Return (ASCII newline) causes a hard line-break, and the following text begins on a new line. A line-break character is inserted in the text, which is drawn as a few pels of white-space (for selection purposes).

**Keystroke commands:** For all the following keys, unless otherwise noted, the display is scrolled, if necessary, to keep the cursor point visible. Where noted, the cursor setting behaves differently in insert mode than in overtype mode. This is subject to the boundary conditions noted above.

|                         |   |
|-------------------------|---|
| <b>Del</b>              | Causes the contents of the selection region to be deleted. If the selection region contains no text, it causes the character to the right of the cursor to be deleted.  |
| <b>Shift+Del</b>        | Causes the contents of the selection region to be cut to the clipboard.   |
| <b>Insert</b>           | Toggles between insert and overtype mode. The MLE ignores the Insert key when it occurs without a modifier.   |
| <b>Shift+Ins</b>        | Causes the contents of the clipboard to replace the selection region.   |
| <b>Ctrl+Ins</b>         | Causes the selection region to be copied to the clipboard. The selection region is not otherwise affected.  |
| <b>Backspace</b>        | Functions similar to Del. If the selection is not empty, Backspace deletes the selection. If the selection is empty, Backspace deletes the character to the left of the cursor point. If the MLE is in overtype mode, the anchor point is set, and the cursor point is moved to be one character previous in the text. If no such character exists (because the anchor is set to the beginning of the text) the cursor is set to the anchor point. If the MLE is in insert mode, the cursor and anchor points are set, as defined at the start of this chapter. |
| <b>Down Arrow</b>       | Sets the cursor point to the closest insertion point on the following line, then sets the anchor point to the cursor point (insertion mode) or one character following (overtype mode).   |
| <b>Shift+Down Arrow</b> | Causes the cursor point to be moved to the closest insertion point on the following line. The anchor point does not move.   |
| <b>Up Arrow</b>         | Sets the cursor point to the closest insertion point on the preceding line, then sets the anchor point to the cursor point (insert mode) or one character following (overtype mode).  |
| <b>Shift+Up</b>         | Sets the cursor point to the closest insertion point on the preceding line. The anchor point is not moved.  |

|                                      |   |
|--------------------------------------|---|
| <b>Right Arrow</b>                   | Sets the cursor point to the insertion point one character following the cursor point. The anchor point is set to the cursor point (insert mode) or one character following (overtyping mode).  |
| <b>Shift+Right</b>                   | Causes the cursor point to be set to the insertion point immediately following the previous cursor point. The anchor point is not moved.  |
| <b>Left and Shift+Left</b>           | Work analogously.   |
| <b>Ctrl+Right</b>                    | Moves the cursor point to the insertion point immediately preceding the next word in the text including trailing spaces, and sets the anchor point to be equal to (insert mode) or one character following (overtyping mode) the cursor point. The EOL (hard line-break) and tab characters are treated as words. |
| <b>Ctrl+Shift+Right</b>              | Moves only the cursor point in the same way as Ctrl+Right, but leaves the anchor point unmoved.   |
| <b>Ctrl+Left</b>                     | Moves the cursor point to the preceding insertion point at the beginning of a word, and sets the anchor point to be equal to (insert mode) or one character following (overtyping mode) the cursor point. The EOL (hard line-break) and tab characters are treated as words.                                      |
| <b>Ctrl+Shift+Left</b>               | Moves only the cursor point in the same way as Ctrl+Left but leaves the anchor point unmoved.   |
| <b>Pagedown and Pageup</b>           | Cause the display to be scrolled one screen at a time in either direction. This behavior is the same as would be encountered during a page-down or page-up caused by the scroll-bar.  |
| <b>Ctrl+Pagedown and Ctrl+Pageup</b> | Cause the display to be scrolled one screen at a time to the right or left respectively. This behavior is the same as would be encountered during a page-right or page-left caused by the scroll-bar.   |
| <b>Home</b>                          | Sets the cursor point to the insertion point at the beginning of the line containing the cursor point, and sets the anchor point equal to (insert mode) or one character following (overtyping mode).   |
| <b>Shift+Home</b>                    | Moves the cursor point to the insertion point at the beginning of the line. The anchor point is not moved.  |
| <b>End</b>                           | Sets the anchor point to the insertion point at the end of the line containing the cursor point. If the last character on the line is a line-break character, the anchor is positioned just before it. The cursor is set equal to (insert mode) or one character previous to (overtyping mode) the anchor.        |

|                        |  |
|------------------------|--|
| <b>Shift+End</b>       | Moves the cursor point to the insertion point at the end of the line, as above. The anchor point is not moved.   |
| <b>Ctrl+Home</b>       | Moves the cursor point to the insertion point at the beginning of the document. The anchor point is set equal to (insert mode) or one character following it (overtyping mode).                  |
| <b>Ctrl+End</b>        | Moves the anchor point to the insertion point at the end of the document. The cursor point is set to be equal to the anchor point (insert mode) or one character preceding it (overtyping mode). |
| <b>Ctrl+Shift+Home</b> | Moves the cursor point in the same way as Ctrl+Home, but leaves the anchor point unmoved.  |
| <b>Ctrl+Shift+End</b>  | Moves the cursor point in the same way as Ctrl+End, but leaves the anchor point unmoved.   |

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_CHAR

---

## WM\_ENABLE (in Multiline Entry Fields)

For the cause of this message, see "WM\_ENABLE" on page 10-43.

For a description of the parameters, see "WM\_ENABLE" on page 10-43.

### Remarks

The multi-line entry field control window procedure responds to this message by setting the enable state and by setting *ulReserved* to 0.

Disabling the window is similar, but not identical, to MLM\_DISABLELREFRESH. Enabling the window is similar, but not identical, to MLM\_ENABLELREFRESH. (Note that this also applies to window styles.) The difference is that a disabled window receives no mouse or keyboard input whereas with MLM\_DISABLELREFRESH it receives the input but discards it.

### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

### Related Messages

- WM\_ENABLE

---

## WM\_MOUSEMOVE (in Multiline Entry Fields)

For the cause of this message, see "WM\_MOUSEMOVE" on page 10-59.

For a description of the parameters, see "WM\_MOUSEMOVE" on page 10-59.

### Remarks

The mouse pointer moves and is of interest to the MLE. If refresh is disabled, the pointer is set to the wait icon (a clock face). If refresh is enabled, the pointer is set to an I-beam. This message can occur during dragging or when simply tracking the mouse.

|                    |   |
|--------------------|---|
| Dragging           | Dragging sets the selection anchor to be the point where dragging begins, and moves the cursor point along with it as the mouse is moved. Moving the pointer into the margins while dragging produces a scroll in the appropriate direction and continues selecting.  |
| Margin Mouse Event | All mouse events in a margin cause the MLE to send a MLN_MARGIN notification to the owner window MLE. This message has, as its parameters, the original mouse message. The owner can process the notification or not. If the owner does not process the message, the event is treated as if it occurred on the closest point in the text. |

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to 0 (FALSE).

### Related Messages

- WM\_MOUSEMOVE

---

## WM\_QUERYWINDOWPARAMS (in Multiline Entry Fields)

This message occurs when an application queries the entry field control window parameters.

For a description of the parameters, see "WM\_QUERYWINDOWPARAMS" on page 10-75.

### Remarks

The multi-line entry field control window procedure responds to this message by returning the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure, identified by the *pwndparams* parameter.

In response to the WPM\_CCHTEXT flag, the text length is reported in the CF\_TEXT format. If it exceeds 64KB-1, then this value is reported. In response to the WPM\_TEXT flag, text up to the amount returned for the WPM\_CCHTEXT value is placed at the indicated location in CF\_TEXT format.

## Default Processing

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to 0 and sets *rc* to FALSE.

## Related Messages

- WM\_QUERYWINDOWPARAMS

---

## WM\_SETWINDOWPARAMS (in Multiline Entry Fields)

This message occurs when an application sets or changes the entry field control window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

## Remarks

The multi-line entry field control window procedure responds to this message by setting the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure, identified by the *pwndparams* parameter.

If the MLE text is to be set by this message, it is assumed to be in CF\_TEXT format (see MLM\_FORMAT) and all existing text is deleted before the new text is inserted. Note that a Control Data structure can be associated with the window parameters, in which case any field in that structure can cause a change to the MLE.

## Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

## Related Messages

- WM\_SETWINDOWPARAMS



---

## Chapter 17. Combination-Box Control Window Processing

This system-provided window procedure processes the actions on a prompted entry field (combination-box) control (WC\_COMBOBOX).

### Purpose

A combination-box consists of an entry field control and a list box control merged into a single control. The list, which is usually limited in size, is displayed below the entry field, and offset one dialog-box unit to its right.

When the combination-box control has the focus, the text in the entry field is given selected emphasis and, if the list box control has a matching entry, it is scrolled to show that match at the top of the list.

A combination-box, while sometimes only showing the entryfield, also owns the area occupied by the invisible list box. Another window can and will be clipped to it if they have clipping flags set.

---

### Combination Box Control Styles

These combination-box control styles are available:

#### **CBS\_SIMPLE**

Both the entry field control and the list box control are visible. When the selection changes in the list box control, the text of the selected item in the list box control is placed in the entry field. Also, the text in the entry field is completed by extending the text of the entry field with the closest match from the list box.

#### **CBS\_DROPDOWN**

Inherits all the properties of a combination-box control with a style of CBS\_SIMPLE and, in addition, the list box control is hidden until the user requests that it should be displayed.

#### **CBS\_DROPDOWNLIST**

In which the entry field control is replaced by a static control, that displays the current selection from the list box control. The user must explicitly cause the display of the list box control in order to make alternative selections in the list box.

---

### Combination Box Control Data

None.



---

## Default Colors

The following system colors are used when the system draws button controls:

SYSCLR\_WINDOWFRAME  
SYSCLR\_ENTRYFIELD  
SYSCLR\_WINDOW  
SYSCLR\_BUTTONMIDDLE  
SYSCLR\_BUTTONDARK  
SYSCLR\_BUTTONLIGHT  
SYSCLR\_OUTPUTTEXT  
SYSCLR\_WINDOWTEXT  
SYSCLR\_HIGHLIGHTFOREGROUND  
SYSCLR\_HIGHLIGHTBACKGROUND  
SYSCLR\_FIELDBACKGROUND  
SYSCLR\_WINDOWFRAME.

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_FOREGROUNDCOLOR  
PP\_DISABLEDFOREGROUND  
PP\_HIGHLIGHTFOREGROUND  
PP\_FONTNAMESIZE  
PP\_BORDERCOLOR.

---

## Combo Box Control Notification Messages

The combo box control uses most of the same window messages as the entry field control and the list box control to notify its owner of significant events.

---

### WM\_CONTROL (in Combination Boxes)

For the cause of this message, see "WM\_CONTROL" on page 10-39.

#### Parameters

##### param1

##### usid (USHORT)

Control window identity.

##### usnotifycode (USHORT)

Notify code.

|              |  |
|--------------|--|
| CBN_EFCHANGE | The content of the entry field control has changed, and the change has been displayed on the screen.   |
| CBN_MEMERROR | The entry field control cannot allocate the storage necessary to accommodate window text of the length implied by the EM_SETTEXTLIMIT message.   |
| CBN_EFSCROLL | The entry field control is about to scroll horizontally. This can happen in these circumstances: <ul style="list-style-type: none"><li>• The application has issued a WinScrollWindow call.</li><li>• The content of the entry field control has changed.</li><li>• The caret has moved.</li></ul> The entry field control must scroll to show the caret position. |
| CBN_LBSELECT | An item in the list box control has been selected.   |
| CBN_LBSCROLL | The list box is about to scroll.   |
| CBN_SHOWLIST | The list box is about to be displayed.   |
| CBN_ENTER    | The user has depressed the ENTER key or double clicked (single clicked in the case of a drop-down list) on an item in the list box control.  |

##### param2

##### hwndcontrolspect (HWND)

Combination (combo) window handle.

**Returns****ulReserved** (ULONG)

Reserved value, should be 0.

**Remarks**

The entry field control window procedure generates this message and sends it to its owner, informing the owner of the event.

**Default Processing**

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

**Related Messages**

- WM\_CONTROL

---

## Combo Box Control Window Messages

The combo box control uses most of the same messages as the entry field control and the list box control. In particular, the following messages are supported to achieve the functions of a combo box. These messages are explained in detail in the entry field control window messages and the list box control window messages sections.

|   |  |
|---|--|
| <b>WM_SETWINDOWPARAMS (in Entry Fields)</b>   | To set the text of the entry field.  |
| <b>WM_QUERYWINDOWPARAMS (in Entry Fields)</b> | To obtain the text of the entry field.   |
| <b>LM_QUERYITEMCOUNT</b>                      | To obtain the count of items in the list box control.  |
| <b>LM_INSERTITEM</b>                          | To insert an item into the list box control.   |
| <b>LM_SETTOPINDEX</b>                         | To scroll the list box control so that the specified item is at the top.   |
| <b>LM_QUERYTOPINDEX</b>                       | To obtain the index of the item at the top of the list box control.  |
| <b>LM_DELETEITEM</b>                          | To delete an item from the list box control. If necessary, this also changes the content of the entry field to the item at the top of the list box control.  |
| <b>LM_SELECTITEM</b>                          | To select a specified item in the list box control. Also, this changes the content of the entry field to the item at the top of the list box control and, if the list box control is not visible, causes the list box control to 'dropdown' below the entry field control. |
| <b>LM_QUERYSELECTION</b>                      | To obtain the current selection in the list box control.   |
| <b>LM_SETITEMTEXT</b>                         | To change the text of an item in the list box control. If necessary, this also changes the content of the entry field control.   |
| <b>LM_QUERYITEMTEXT</b>                       | To obtain the text of an item in the list box control.   |
| <b>LM_QUERYITEMTEXTLENGTH</b>                 | To obtain the length of the text of an item in the list box control.   |
| <b>LM_SEARCHSTRING</b>                        | To obtain the index of an item in the list box control containing a specified string.  |
| <b>LM_DELETEALL</b>                           | To delete all the items in the list box control.   |
| <b>WM_ENABLE</b>                              | To enable the combo box control to respond to input.   |
| <b>EM_QUERYFIRSTCHAR</b>                      | To obtain the character displayed at the left edge of the entry field control.   |
| <b>EM_SETFIRSTCHAR</b>                        | To scroll the entry field control so that the specified character is displayed at the left edge of the entry field control.  |
| <b>EM_QUERYCHANGED</b>                        | To obtain the changes to the entry field control.  |

|                        |   |
|------------------------|---|
| <b>EM_QUERYSEL</b>     | To obtain the current selection of the entry field control.   |
| <b>EM_SETSEL</b>       | To set the current selection of the entry field control.  |
| <b>EM_SETTEXTLIMIT</b> | To set the maximum number of characters to be contained in the entry field control.   |
| <b>EM_CUT</b>          | To place the contents of the selection of the entry field control into the clipboard and then delete those contents from the entry field control. |
| <b>EM_PASTE</b>        | To place the contents of the clipboard into the entry field control.  |
| <b>EM_COPY</b>         | To place the contents of the selection of the entry field control into the clipboard.   |
| <b>EM_CLEAR</b>        | To clear the current selection of the entry field control.  |

This section describes the combo box control window procedure actions on receiving these messages:

---

## **CBM\_HILITE**

This message sets the highlighting state of the entry field control.

### **Parameters**

**param1**

**usHilite** (USHORT)

Highlighting indicator.

TRUE Highlight the entry field control.

FALSE Do not highlight the entry field control.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Returns**

**rc** (BOOL)

Changed indicator.

TRUE The highlighting state of the entry field has been changed.

FALSE The highlighting state of the entry field has not been changed.

### **Remarks**

The combo box control window procedure responds to this message by setting the highlighting state of the entry field control.

## Default Processing

WinDefWindowProc does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## CBM\_ISLISTSHOWING

This message determines if the list box control is showing.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Showing indicator.

TRUE The list box control is showing.

FALSE The list box control is not showing.

### Remarks

The combo box control window procedure responds to this message by indicating if the list box control is showing.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## CBM\_SHOWLIST

This message sets the showing state of the list box control.

### Parameters

**param1**

**usShowing** (USHORT)

Showing indicator.

TRUE Show the list box control.

FALSE Do not show the list box control.

## **param2**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **rc (BOOL)**

Changed indicator.

TRUE    The list box showing state has been changed.

FALSE   The list box showing state has not been changed.

## **Remarks**

The combo box control window procedure responds to this message by setting the showing state of the list box control.

This message has no effect on a combo box control whose style is `CBS_SIMPLE`.

Hiding the list box control has no effect on the selection in the list box control. The selection in the list box control must be changed by the use of a `LM_SELECTITEM` message.

## **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set `rc` to the default value of `FALSE`.

---

## Chapter 18. Scroll Bar Control Window Processing

This system-provided window procedure processes the actions on a scroll bar control (WC\_SCROLLBAR).

### Purpose

Scroll bars are controls used to indicate that additional information can be displayed in a window, logically to the left or right for horizontal scroll bars, logically above or below for vertical scroll bars. The user interface for scroll bars allows for scrolling one unit or one page at a time, or alternatively picking up the scroll bar slider and moving it to a position in the scroll bar that indicates a logical position in the data.

---

### Scroll Bar Control Styles

These scroll bar control styles are available:

|                      |   |
|----------------------|---|
| <b>SBS_HORZ</b>      | Create a horizontal scroll bar.   |
| <b>SBS_VERT</b>      | Create a vertical scroll bar.   |
| <b>SBS_THUMBSIZE</b> | Indicates the presence of the <i>cVisible</i> and <i>cTotal</i> parameters in the SBCDATA data structure. |
| <b>SBS_AUTOTRACK</b> | The slider scrolls as more information is being displayed on the screen.                                  |
| <b>SBS_AUTOSIZE</b>  | The scroll bar slider changes size to reflect the amount of data contained in the window.                 |

---

### Scroll Bar Control Data

See "SBCDATA" on page A-182.

---

### Default Colors

The following system colors are used when the system draws button controls:

SYSCLR\_SCROLLBAR  
SYSCLR\_WINDOWFRAME  
SYSCLR\_FIELDBACKGROUND  
SYSCLR\_WINDOW  
SYSCLR\_BUTTONMIDDLE.



Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_FOREGROUND\_COLOR  
PP\_BORDER\_COLOR  
PP\_HILITE\_FOREGROUND\_COLOR.

---

## Scroll Bar System Values

Applications can use the following system values to create and add control scroll bars:

|                           |  |
|---------------------------|--|
| <b>SV_CXVSCROLL</b>       | Width of the vertical scroll-bar.  |
| <b>SV_CYHSCROLL</b>       | Height of the horizontal scroll-bar.   |
| <b>SV_CVSCROLLARROW</b>   | Height of the vertical scroll-bar arrow bit maps.  |
| <b>SV_CXHSCROLLARROW</b>  | Height of the horizontal scroll-bar arrow bit maps.  |
| <b>SV_FIRSTSCROLLRATE</b> | The delay (in milliseconds) before autoscrolling starts, when using a scroll bar.  |
| <b>SV_SCROLLRATE</b>      | The delay (in milliseconds) between scroll operations, when using a scroll bar.  |
| <b>SYSCLR_SCROLLBAR</b>   | Color for drawing scroll-bar backgrounds.  |
| <b>TID_SCROLL</b>         | Timer ID for a reserved scrolling time. This is used for sending notification messages when a scroll-arrow or scroll-bar background is selected. |

---

## Scroll Bar Control Notification Messages

These messages are initiated by the scroll bar control window procedure to notify its owner of significant events.

---

### WM\_HSCROLL (in Horizontal Scroll Bars)

For the cause of this message, see “WM\_HSCROLL” on page 10-51.

For a description of the parameters, see “WM\_HSCROLL” on page 10-51.

#### Remarks

The scroll bar control window procedure generates this message and posts it to its owner, informing the owner of the event.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_HSCROLL

---

### WM\_VSCROLL (in Vertical Scroll Bars)

For the cause of this message, see “WM\_VSCROLL” on page 10-99.

For a description of the parameters, see “WM\_VSCROLL” on page 10-99.

#### Remarks

The scroll bar control window procedure generates this message and posts the message to the owner of the procedure, informing the owner of the event.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_VSCROLL

---

## Scroll Bar Control Window Messages

This section describes the scroll bar control window procedure actions on receiving the following messages.

---

### SBM\_QUERYPOS

This message returns the current slider position in a scroll bar window.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**slider** (SHORT)

Slider position.

#### Remarks

The scroll bar control window procedure responds to this message by returning the current slider position.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *slider* to the default value of 0.

---

### SBM\_QUERYRANGE

This message returns the scroll bar range minimum and maximum values.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## **Returns**

**ReturnCode**

**sfirst** (SHORT)

First bound.

**slast** (SHORT)

Last bound.

## **Remarks**

The scroll bar control window procedure responds to this message by returning the first and last bounds of the scroll bar range.

## **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *ReturnCode* to the default value of *sfirst* and *slast* to 0.

---

## **SBM\_SETPOS**

This message sets the position of the slider in a scroll bar window.

## **Parameters**

**param1**

**sslider** (SHORT)

Position of slider.

If this value is outside the scroll-bar range, the slider is moved to the nearest valid position within the range.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred

## Remarks

The scroll bar control window procedure responds to this message by setting the position of the slider.

The scroll bar control is redrawn to reflect the change.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it.

---

## SBM\_SETSCROLLBAR

This message sets the scroll-bar range and slider position.

## Parameters

**param1**

**slider** (SHORT)

Position of slider.

If this value is outside the scroll-bar range, the slider is moved to the nearest valid position within the range.

**param2**

**sfirst** (SHORT)

First bound.

This value must not be less than 0. If a value less than 0 is supplied, 0 is used as the value.

**slast** (SHORT)

Last bound.

The value must not be less than 0 or *sfirst*. If a value less than this is supplied, the higher of 0 or *sfirst* is used as the value.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The scroll bar control window procedure responds to this message by setting the values of the information range and the position of the slider.

The scroll bar is redrawn to reflect the change.

For example, if a scroll-bar is to allow scrolling through 100 lines of text, of which 50 are visible at any one time, and the top display line is currently number 25, *sfirst* should be set to 1, *slast* to 51 (since there are only 51 positions at which the slider may be placed), and *slider* to 25. The SBM\_SETTHUMBSIZE message should be used in this example to set the slider size to 50 visible parts out of 100.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it.

---

## SBM\_SETTHUMBSIZE

This message sets the scroll bar slider size.

### Parameters

**param1**

**svisible** (SHORT)

Size of the visible part of the document.

**stotal** (SHORT)

Size of the entire document.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The scroll bar control window procedure responds to this message by setting the size of the slider proportional to the visible part of the document. If the visible part exceeds or is equal to the entire document the scroll bar is disabled, otherwise the scroll bar is enabled.

The scroll bar is redrawn to reflect the change.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it.

---

## WM\_QUERYCONVERTPOS (in Scroll Bars)

For the cause of this message, see "WM\_QUERYCONVERTPOS" on page 10-72.

For a description of the parameters, see "WM\_QUERYCONVERTPOS" on page 10-72.

## Remarks

The scroll bar control window procedure returns QCP\_NOCONVERT.

## Default Processing

For the default window procedure processing of this message see "WM\_QUERYCONVERTPOS" on page 10-72.

## Related Messages

- WM\_QUERYCONVERTPOS

---

## WM\_QUERYWINDOWPARAMS (in Scroll Bars)

This message occurs when an application queries the scroll bar control window parameters.

For a description of the parameters, see "WM\_QUERYWINDOWPARAMS" on page 10-75.

## Remarks

The scroll bar control window procedure responds to this message by returning the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure identified by the *pwndparams* parameter.

## Default Processing

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to 0 and sets *rc* to FALSE.

### Related Messages

- WM\_QUERYWINDOWPARAMS

---

## WM\_SETWINDOWPARAMS (in Scroll Bars)

This message occurs when an application sets or changes the scroll bar control window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### Remarks

The scroll bar control window procedure responds to this message by setting the window parameters indicated by the *fsStatus* parameter of the WNDPARAMS data structure identified by the *pwndparams* parameter.

### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### Related Messages

- WM\_SETWINDOWPARAMS





---

## Chapter 19. Spin Button Control Window Processing

This system-provided window procedure processes the actions on a spin button control (WC\_SPINBUTTON).

---

### Purpose

A spin button control (WC\_SPINBUTTON window class) is a visual component whose specific purpose is to give users quick access to a finite set of data. The spin button allows users to select from a scrollable ring of choices. Since users can see only one item at a time, the spin button control should be used only with data that is intuitively related, such as a list of months of the year, or an alphabetic list of cities or states.

A spin button consists of at least one spin field that is a single-line entry field (SLE), and up and down arrows that are stacked on top of one another. These arrows are positioned at the right of the SLE.

You can create multifield spin buttons for those applications in which users must select more than one value. For example, in setting a date the spin button control can provide individual fields for setting the month, day, and year. The first spin field in the spin button could contain a list of months, the second spin field could contain a list of numbers and the third spin field could contain a list of years.

---

### Spin Button Control Styles

Create a spin button using the style bits listed below. These styles can be joined together by using logical ORs (|).

- Specify one of the following to determine whether a spin field will be a master or a servant. If neither is specified, SPBS\_SERVANT is the default.

**SPBS\_MASTER**

The spin button component consists of at least one single line entry field (SLE), or spin field, and two arrows, the Up Arrow and the Down Arrow. When a spin button contains more than one spin field, the master component contains the spin arrows. If the component contains only one spin field, it should be a master.

**SPBS\_SERVANT**

You can create a multifield spin button by spinning servants from the master.

- Specify one of the following to determine the type of characters allowed in the spin field:

**SPBS\_ALLCHARACTERS**

Any character can be typed in the spin field. This is the default.

**SPBS\_NUMERICONLY**

Only the digits 0—9 and the minus sign (-) can be typed in the spin field.

**SPBS\_READONLY**

Nothing can be typed in the spin field.

- Specify one of the following to determine how the text is to be presented in the spin field:

**SPBS\_JUSTLEFT**            Left-justify the text. This is the default.  
**SPBS\_JUSTRIGHT**        Right-justify the text.  
**SPBS\_JUSTCENTER**      Center the text.

- Specify the following when you do not want a border around the spin button:

**SPBS\_NOBORDER**        Suppresses drawing a border.

- Specify the following to increase the spin speed:

**SPBS\_FASTSPIN**        Enables the spin button to increase the spin speed with time. The speed doubles every two seconds.

**Note:** The spin button skips information when this option is specified. Do not use SPBS\_FASTSPIN if the application requires that this field be checked each time a spin up or spin down occurs. Do not specify this option on a master component that has servants spun from it.

- Specify the following to pad numeric fields with 0s. This is useful when the spin field contains values that represent time or money.

**SPBS\_PADWITHZEROS**    The output number is padded at the front between the first non-zero digit and the field width, or 11 characters, whichever is the lesser. The negative sign, if there is one, is retained. The maximum number of characters required to display a LONG number is 11.

---

## Spin Button Control Data

See SPBCDATA

---

## Spin Button Control Notification Message

This message is initiated by the spin button control window to notify its owner of significant events.

---

### WM\_CONTROL (in Spin Button Controls)

For the cause of this message, see “WM\_CONTROL” on page 10-39.

#### Parameters

##### param1

###### id (USHORT)

Identity of the spin button component window.

###### notifycode (USHORT)

Notification code.

|                |  |
|----------------|--|
| SPBN_UPARROW   | Tells the application that the Up Arrow was clicked on, or the Up Arrow key was pressed.                         |
| SPBN_DOWNARROW | Tells the application that the Down Arrow was clicked on, or the Down Arrow key was pressed.                     |
| SPBN_SETFOCUS  | Tells the application which spin field was selected.   |
| SPBN_KILLFOCUS | Tells the application when the spin field loses focus.   |
| SPBN_ENDSPIN   | Tells the application that the user released the select button or one of the arrow keys while spinning a button. |
| SPBN_CHANGE    | Tells the application that the contents of the spin field changed.   |

##### param2

###### hwnd (HWND)

Window handle.

The interpretation of this handle is dependent upon the following notification codes:

- SPBN\_UPARROW, SPBN\_DOWNARROW, and SPBN\_ENDSPIN.  
The *param2* parameter is the handle to the currently selected spin field in a particular master-servant setup. If either the Up or Down Arrow is clicked on and none of a spin button’s servants are currently selected, the master will return a handle to itself.
- SPBN\_SETFOCUS  
The *param2* parameter is the handle of the currently selected spin field.  
This message tells the application which spin field is selected.

- SPBN\_KILLFOCUS

The *param2* parameter is NULLHANDLE if the spin field loses focus or no spin field is currently selected.

This message tells the application when a spin field loses focus.

**Note:** Both SPBN\_KILLFOCUS and SPBN\_SETFOCUS are set independently. You must check this message only when the application does not specify a master-servant relationship.

- SPBN\_CHANGE

The *param2* parameter is the handle of the spin button in which the spin field text changed.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is sent when, as specified by *notifycode*, the spin button component must tell its owner of a significant event.

### Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return 0.

---

## Spin Button Control Window Messages

This section describes the spin button control window procedure actions on receiving the following messages.

---

## SPBM\_OVERRIDESETLIMITS

This message causes the component to set or reset numeric limits.

### Parameters

**param1**

**IUpLimit** (LONG)

Upper limit.

**param2**

**ILowLimit** (LONG)

Lower limit.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

## Remarks

The application sends this message to the component to set or reset numeric limits.

This message is functionally identical to `SPBM_SETLIMITS`, except that the current value of the spin button does not change if it is out of range.

When the upper limit is less than the lower limit, `FALSE` is returned.

## Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return `FALSE`.

---

## SPBM\_QUERYLIMITS

This message enables an application to query the limits of a numeric spin field.

## Parameters

**param1**

**pIUpLimit** (PLONG)

Pointer to a LONG that will receive the returned upper limit.

**param2**

**pILowLimit** (PLONG)

Pointer to a LONG that will receive the returned lower limit.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The application sends this message to the component to determine the limits of a numeric spin field.

When the spin button has no data, or when it is spinning an array, `FALSE` is returned.

## Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE

---

## SPBM\_QUERYVALUE

This message causes the component to show the value in the spin field.

### Parameters

#### param1

##### pStorage (PVOID)

Place for returned value.

A place for the returned value. This value is either the address of a string or the address of a long variable.

If the *usBufSize* is 0, *param1* is assumed to be an address of a long variable.

If *param1* is Other, it is assumed to be an address of a string.

NULL Causes the spin button to process the reset or update as specified, but it will not try to return a value to the application.

Other The address where the value is returned.

#### param2

##### usBufSize (USHORT)

Buffer size.

If *usBufSize* is too small to return all of the text, the spin button returns as much of the text as it can.

0 The spin button assumes that *param1* is the address of a long variable. If the data in the spin button is spinning between an upper and lower limit, the current value is passed back in the variable.

If the data in the spin button is in an array, the index of the current array value (or last valid value) is passed back in the variable.

Other The spin button assumes that *param1* is the address of a string. The information passed back in the string is dependent upon the flags in the *usValue* parameter.

## usValue (USHORT)

Update/reset value.

Controls how the spin field is updated.

**SPBQ\_UPDATEIFVALID** Update the contents of the spin field if the value is valid. This is the default.

Specifying this flag on a query will *not* update the contents of the spin field if it is *exactly* the same as an item in the spin button list.

If an item in the list is Monday, specifying **SPBQ\_UPDATEIFVALID** updates the spin field contents when **MONDAY**, **monday**, or **mONDAY** are typed, but not when **Monday** is typed. This prevents recursion if the application checks for the validity each time a **SPBN\_CHANGE** message is sent from the component.

**SPBQ\_ALWAYSUPDATE** Update the contents of the spin field if the value is valid. Reset the contents of the spin field to the last valid value if the field contains data that is not valid.

If the spin button is spinning numbers between an upper and a lower limit, and the content of the spin field is a valid number that is out of range, the spin button does not reset itself to the last valid value. It sets the current position at the upper limit when the out-of-range number specified is above the upper limit. It sets the current position at the lower limit when the out-of-range number is below the lower limit.

When the current value is changed, the return of the query message is still **FALSE**.

**SPBQ\_DONOTUPDATE** Do not update the contents of the spin field, even if the value is valid.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Successful completion.

**FALSE** Error occurred.

## Remarks

The application sends this message to the component to determine what value is in the spin field. The application sets up a field for the component to deposit the value, and sets a flag to determine what the function does when the value matches or does not match the given spin-list values.

**TRUE** is returned when a matched value is found, or the data is in the range.



FALSE is returned when no match is found, the value is out of range, or no spin data exists.

### Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SETARRAY

This message causes the component to set or reset the array of data.

### Parameters

**param1**

**pStr1** (PSZ)

Pointer to the new array of values.

**param2**

**usItems** (USHORT)

Number of items in the array.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Remarks

The application sends this message to the component to set or reset the array of data.

The component tries to leave the current value unchanged. However, if the current value is out of range for the new array, it is moved to the closest extreme. Thus, if the current value is less than 0, it is moved to 0. If the current value is greater than the previous value, it is set to the previous value.

If the data exceeds 64KB, or if *param1* or *param2* equal 0, FALSE is returned.

### Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SETCURRENTVALUE

This message causes the component to set or reset the current numeric value or array index.

### Parameters

**param1**

**IValue (LONG)**

Array value or index.

Current value or index of array.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### Returns

**rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE Error occurred.

### Remarks

The application sends this message to the component to set or reset the current numeric value or array index.

FALSE is returned when the value is out of range or there is no spin data.

### Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SETLIMITS

This message causes the component to set or reset numeric limits.

### Parameters

**param1**

**IUpLimit (LONG)**

Upper limit.

## **param2**

### **lLowLimit (LONG)**

Lower limit.

### **rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## **Remarks**

The application sends this message to the component to set or reset numeric limits. The component sets the current value to the content in the spin field when it is a valid number. When the current value is out of the range of the limits, it is moved to the nearest limit, upper or lower.

If the upper limit is less than the lower limit, FALSE is returned.

## **Default Processing**

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## **SPBM\_SETMASTER**

This message causes the component to identify its master.

## **Parameters**

### **param1**

#### **hwnd (HWND)**

Handle of master component.

### **param2**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **rc (BOOL)**

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The application sends this message to the component to tell a component who its master is.

When the application wants to take control of the spin button, it must set the *param1* of each spin button to NULLHANDLE. This must be done, for example, when a spin button with a non-contiguous list of spin values is created (2, 4, 6, 8, 10...). When the *param1* of a spin button is NULLHANDLE, the spin button does not perform the following default functions:

- Spin up or down on its own when the Up or Down Arrow key is pressed.
- Spin up or down when the Up or Down Arrow of the master is pressed.
- A master does not take the focus when its arrows are pressed and none of its servants have focus.
- The spin button does not send itself an SPBM\_QUERYVALUE message with the SPBQ\_ALWAYSUPDATE flag to update the current value when an SPBM\_SPINUP or SPBM\_SPINDOWN message is received.
- The spin button does not fast spin.

## Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SETTEXTLIMIT

This message sets the maximum number of characters allowed in a spin field.

### Parameters

#### param1

**usLimit** (USHORT)

Character limit.

Number of characters to allow.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The application sends this message to set the maximum number of characters allowed in the spin field. The size limit of the spin field is 255 characters. This is the default.

When the size exceeds 255 characters, FALSE is returned,

## Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SPINDOWN

This message causes the component to show the previous value (spin backward).

## Parameters

**param1**

**ulItem** (ULONG)

Number of values to spin down.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The application sends this message to the component when it wants the previous value shown (spin backward).

When there is no data to spin, FALSE is returned.

## Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.

---

## SPBM\_SPINUP

This message causes the component to show the next value (spin forward).

### Parameters

#### param1

**ullItem** (ULONG)

Number of values to spin up.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

TRUE     Successful completion

FALSE    Error occurred.

### Remarks

The application sends this message to the component when it wants the next value shown (spin forward).

When there is no data to spin, FALSE is returned.

### Default Processing

The default window procedure does not expect to receive this message and takes no action other than to return FALSE.



---

## Chapter 20. Static Control Window Processing

This system-provided window procedure processes the actions on a static control (WC\_STATIC).

### Purpose

Static controls are simple text fields, bit maps, icons, and boxes that can be used to label or box other controls. Static controls do not accept user input, nor do they send notification messages to their owner.

---

### Static Control Styles

These static control styles are available:

#### SS\_TEXT

Creates a box with formatted text. The text is formatted before it is displayed according to the setting of these text drawing-style flags:

|                  |                      |
|------------------|----------------------|
| <b>DT_LEFT</b>   | Left-justified text  |
| <b>DT_CENTER</b> | Centered text        |
| <b>DT_RIGHT</b>  | Right-justified text |

ORed with one of:

|                   |  |
|-------------------|--|
| <b>DT_TOP</b>     | Text is aligned to top of window               |
| <b>DT_VCENTER</b> | Text is aligned vertically in center of window |
| <b>DT_BOTTOM</b>  | Text is aligned to bottom of window            |

The following text drawing style can also be ORed, but only if DT\_TOP and DT\_LEFT are also specified:

|                     |   |
|---------------------|---|
| <b>DT_WORDBREAK</b> | Text is multi-line with word-wrapping at ends of lines. |
|---------------------|---|

**Note:** For “static” text that can be selected, a Button Control with a style of BS\_NOBORDER can be used.

#### SS\_GROUPBOX

A group box static control is a box that has an identifying text string in its upper left corner. Group boxes are used to collect a group of radio buttons or other controls into a single unit.

#### SS\_ICON

Draws an icon. The text of the static control is a string that is used to derive the resource ID from which the icon is loaded. The format of the string is:

- The first byte is 0xFF, the second byte is the low byte of the resource ID, and the third byte is the high byte of the resource ID.
- The first character is “#”; subsequent characters make up the decimal text representation of the resource ID. This



format can be used for specifying a system icon in a resource file. The decimal string is the value of the appropriate SPTR\_\* constant

If the string is empty or does not follow the format above, no resource is loaded.

The resource is assumed to reside in the resource file of the current process.

This control is resized to the size of the icon.

## **SS\_SYSICON**

This style is the same as SS\_ICON except that the icon ID is specified as one of the system pointer ID values (SPTR\_\* values) rather than a resource ID. This style provides a convenient way to include system icons in application dialog boxes.

## **SS\_BITMAP**

Draws a bit map. The text of the static control names the bit-map resource, as for SS\_ICON.

## **SS\_FGNDRECT**

Creates a rectangle filled with the color of the foreground.

## **SS\_BKGNDRECT**

Creates a rectangle filled with the color of the background.

## **SS\_FGNDFRAME**

Creates a box with frame color equal to the foreground color.

## **SS\_BKGNDFRAME**

Creates a box with frame color equal to the background color.

## **SS\_HALFTONERECT**

Creates a rectangle filled with halftone shading.

## **SS\_HALFTONEFRAME**

Creates a box with halftone shading frame.

## **SS\_AUTOSIZE**

The static control will be sized to make sure the contents fit.

---

## **Static Control Data**

None.

---

## **Default Colors**

The following system colors are used when the system draws button controls:

SYSCLR\_WINDOWFRAME  
SYSCLR\_WINDOWSTATICTEXT  
SYSCLR\_WINDOW  
SYSCLR\_BACKGROUND.

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

PP\_BORDERCOLOR  
PP\_FOREGROUNDCOLOR.

---

## Static Control Notification Messages

No notification messages are initiated by the static control window procedure.

---

## Static Control Window Messages

This section describes the static control window procedure actions on receiving the following messages.

---

### SM\_QUERYHANDLE

This message returns the icon or bit-map handle of a static control.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**hbmHandle** (HBITMAP)

Icon or bit-map handle of the static control.

**NULLHANDLE** No icon or bit-map handle of the static control exists, or an error occurred.

**Other** Icon or bit-map handle of the static control.

#### Remarks

The static control window procedure responds to this message by setting *hbmHandle* to the handle of the icon or bit-map of the static control.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *hbmHandle* to the default value of **NULLHANDLE**.

---

## SM\_SETHANDLE

This message sets the icon or bit-map handle of a static control.

### Parameters

#### param1

##### **hbmHandle** (HBITMAP)

Icon or bit-map handle of a static control.

This is an icon handle when sent to a control with a style of `SS_ICON` or `SS_SYSICON`, and a bit-map handle when sent to a control with a style of `SS_BITMAP`.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **hbmHandle** (HBITMAP)

Icon or bit-map handle of the static control.

`NULLHANDLE` No icon or bit-map handle of the static control exists, or an error occurred.

Other Icon or bit-map handle of the static control.

### Remarks

The static control window procedure responds to this message by setting the icon or bit-map handle of a static control to the value specified by *hbmHandle*, and causes the static control to be redrawn, using the new item handle.

It should only be sent to a control with a style of `SS_BITMAP`, `SS_ICON`, or `SS_SYSICON`.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *hbmHandle* to the default value of `NULLHANDLE`.

---

## WM\_MATCHMNEMONIC (in Static Controls)

For the cause of this message, see “`WM_MATCHMNEMONIC`” on page 10-55.

For a description of the parameters, see “`WM_MATCHMNEMONIC`” on page 10-55.

### Remarks

The static control window procedure responds to this message by setting *rc* as appropriate.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_MATCHMNEMONIC

---

## **WM\_QUERYCONVERTPOS (in Static Controls)**

For the cause of this message, see “WM\_QUERYCONVERTPOS” on page 10-72.

For a description of the parameters, see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Remarks**

The static control window procedure returns QCP\_NOCONVERT.

### **Default Processing**

For the default window procedure processing of this message see “WM\_QUERYCONVERTPOS” on page 10-72.

### **Related Messages**

- WM\_QUERYCONVERTPOS

---

## **WM\_QUERYWINDOWPARAMS (in Static Controls)**

This message occurs when an application queries the static control window procedure window parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### **Remarks**

The static control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure sets the *cchText*, *cbPresParams*, and *cbCtlData* parameters of the WNDPARAMS data structure, identified by *pwndparams*, to zero and sets *rc* to FALSE.

### **Related Messages**

- WM\_QUERYWINDOWPARAMS

---

## **WM\_SETWINDOWPARAMS (in Static Controls)**

This message occurs when an application sets or changes the static control window procedure window parameters.

For a description of the parameters, see "WM\_SETWINDOWPARAMS" on page 10-86.

### **Remarks**

The static control window procedure responds to this message by passing it to the default window procedure.

### **Default Processing**

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

### **Related Messages**

- WM\_SETWINDOWPARAMS



---

## Chapter 21. Title Bar Control Window Processing

This system-provided window procedure processes the actions on a title bar control (WC\_TITLEBAR).

### Purpose

The title bar control is the frame control that is used to display the application window title. It is also used to display the active or inactive status of the frame window.

The title bar control also implements the user interface for moving the frame window.

The standard identifier for a title bar control in a frame window is FID\_TITLEBAR.

---

### Title Bar Control Styles

There is only one title bar style, the default.

---

### Title Bar Control Data

None.

---

### Default Colors

The following system colors are used when the system draws button controls:

- SYSCLR\_ACTIVETITLETEXTBGND
- SYSCLR\_ACTIVETITLE
- SYSCLR\_ACTIVETITLETEXT
- SYSCLR\_ACTIVETITLETEXTBGND
- SYSCLR\_INACTIVETITLE
- SYSCLR\_INACTIVETITLETEXT
- SYSCLR\_INACTIVETITLETEXTBGND
- SYSCLR\_TITLEBOTTOM
- SYSCLR\_(IN)ACTIVETITLETEXTBGND
- SYSCLR\_(IN)ACTIVETITLE.

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

- PP\_FONTNAMESIZE
- PP\_ACTIVECOLOR
- PP\_INACTIVECOLOR
- PP\_ACTIVETEXT\*COLOR
- PP\_INACTIVETEXT\*COLOR
- PP\_ACTIVETEXTFGNDCOLOR
- PP\_INACTIVETEXTFGNDCOLOR
- PP\_BORDERCOLOR.



---

## Title Bar Control Notification Messages

These messages are initiated by the title bar control to notify its owner of significant events.

---

### WM\_SYSCOMMAND (in Title Bar Controls)

For the cause of this message, see "WM\_SYSCOMMAND" on page 10-91.

For a description of the parameters, see "WM\_SYSCOMMAND" on page 10-91.

The title bar control window procedure sets *uscmd* to the title bar control identity and *ussource* to CMDSRC\_OTHER.

#### Remarks

The title bar control window procedure generates this message when a mouse input message is received. The window procedure posts the message to the queue of the window owner.

The purpose of this message is to notify the owner window to maximize or restore depending on its current state.

#### Default Processing

The default window procedure takes no action on this message, other than to set *ulReserved* to 0.

#### Related Messages

- WM\_COMMAND

---

### WM\_TRACKFRAME (in Title Bar Controls)

For the cause of this message, see "WM\_TRACKFRAME" on page 10-95.

For a description of the parameters, see "WM\_TRACKFRAME" on page 10-95.

#### Remarks

The title bar control window procedure generates this message and sends it to its owner, informing the owner that a mouse button down message has been received.

#### Default Processing

The default window procedure takes no action on this message, other than to set *rc* to FALSE.

#### Related Messages

- WM\_QUERYTRACKINFO

---

## Title Bar Control Window Messages

This section describes the title bar control window procedure actions on receiving the following messages.

---

### TBM\_QUERYHILITE

This message returns the highlighting state of a title-bar control.

#### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**rc** (BOOL)

Highlighting state.

TRUE Title-bar control is highlighted

FALSE Title-bar control is not highlighted.

#### Remarks

The title bar control window procedure responds to this message by returning the highlighting state of the title-bar window.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## TBM\_SETHILITE

This message is used to highlight or unhighlight a title-bar control.

### Parameters

#### param1

##### **usHighlighted** (USHORT)

Highlighting indicator.

TRUE     Highlight the title-bar control  
FALSE    Remove highlight from the title-bar control.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE     Successful completion  
FALSE    Error occurred.

### Remarks

The title bar control window procedure responds to this message by setting the highlighting state according to *usHighlighted*. If the title bar highlighting state is changed by this message, the title bar will repaint.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it, other than to set *rc* to the default value of FALSE.

---

## WM\_QUERYCONVERTPOS (in Title Bar Controls)

For the cause of this message, see "WM\_QUERYCONVERTPOS" on page 10-72.

For a description of the parameters, see "WM\_QUERYCONVERTPOS" on page 10-72.

### Remarks

The title bar control window procedure returns QCP\_NOCONVERT.

### Default Processing

For the default window procedure processing of this message see "WM\_QUERYCONVERTPOS" on page 10-72.

## Related Messages

- WM\_QUERYCONVERTPOS

---

## WM\_QUERYWINDOWPARAMS (in Title Bars)

This message occurs when an application queries the title bar control window procedure window parameters.

For a description of the parameters, see “WM\_QUERYWINDOWPARAMS” on page 10-75.

### Default Processing

The title bar control window procedure queries the appropriate window parameters in accordance with *pwndparams* and sets *rc* to TRUE if the operation is successful, otherwise to FALSE.

## Related Messages

- WM\_QUERYWINDOWPARAMS

---

## WM\_SETWINDOWPARAMS (in Title Bar Controls)

This message occurs when an application sets or changes the title bar control window procedure window parameters.

For a description of the parameters, see “WM\_SETWINDOWPARAMS” on page 10-86.

### Default Processing

The title bar control window procedure sets the appropriate window parameters in accordance with *pwndparams* and sets *rc* to TRUE if the operation is successful, otherwise to FALSE.

## Related Messages

- WM\_SETWINDOWPARAMS



---

## Chapter 22. Container Control Window Processing

This system-provided window procedure processes the actions on a container control (WC\_CONTAINER).

---

### Purpose

A container control is a visual component whose specific purpose is to hold objects. These objects, or container items, can be anything that either your application or a user might store in a container. Examples are executable programs, word processing files, graphics images, and database records.

Container item data is stored in RECORDCORE or MINIRECORDCORE data structures. Both the application and the container have access to the data stored in these records. See "RECORDCORE" on page A-175 and "MINIRECORDCORE" on page A-124 for descriptions of these data structures.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

The maximum number of records is limited by the amount of memory in the user's computer. The container control does not limit the number of records that a container can have.

The following list shows which types of data can be displayed for each container view. Refer to the description of the container control in the *OS/2 Programming Guide* for more information about the types of views.

| <b>View Types</b>   | <b>Data</b>   |
|---------------------|---|
| <b>Icon view</b>    | Icons or bit maps with text strings beneath                 |
| <b>Name view</b>    | Icons or bit maps with text strings to the right            |
| <b>Text view</b>    | Text strings  |
| <b>Tree view</b>    | Icons or bit maps, and text strings                         |
| <b>Details view</b> | Icons or bit maps, text strings, numbers, times, and dates. |

Direct editing of container item text is supported in all views, including blank text fields.

The container control is designed according to the Common User Access (CUA) guidelines. For example, the CUA direct manipulation protocol is fully supported, enabling a user to visually drag an object in a container window and drop it on another object or container window. In addition, the container control supports CUA-defined selection types and techniques for selecting container items, as well as selection mechanisms, such as pointing devices and the keyboard, and multiple forms of emphasis. For a complete description of CUA containers, refer to the *SAA CUA Guide to User Interface Design* and to the *SAA CUA Advanced Interface Design Reference*.

The container control automatically provides or enables either horizontal or vertical scroll bars, or both, whenever all or part of one or more container items are not visible in a container window's client area.

---

## Container Control Window Words

The container control reserves 4 bytes in its window words for application use. This memory can be accessed using the `WinSetWindowULong` and `WinQueryWindowULong` functions at offset `QWL_USER`.

---

## Container Control Styles and Selection Types

Containers are `WC_CONTAINER` class windows that have the following `CCS_container` styles and selection types. Container control styles and selection types are specified when the container control is created.

### Container Control Styles

The following list defines container style bits that your application can use. These style bits must be set by your application.

#### **CCS\_AUTOPOSITION**

Automatic positioning, which causes container items displayed in the icon view to be arranged when any of the following occur:

- The window size changes
- Container items are inserted, removed, sorted, invalidated, or filtered
- The font or font size changes
- The window title text changes.

In all of these cases, container items are arranged the same as when the `CM_ARRANGE` message is sent. The `CCS_AUTOPOSITION` style bit is valid only when it is used with the icon view (`CV_ICON`).

#### **CCS\_MINIRECORDCORE**

A record style bit that causes the container to interpret all container records as being smaller than they would otherwise be. If a `CM_ALLOCORECORD` message is received, all records are interpreted and allocated according to the information in the `MINIRECORDCORE` data structure instead of the `RECORDCORE` data structure, which is used if this style bit is not specified.

#### **CCS\_READONLY**

A read-only style bit for an entire container, which prevents a user from editing any of the text in a container window. If you do not set this style bit, a user can edit any of the text in a container window unless you set the following read-only attributes in the appropriate data structures:

#### **CA\_TITLEREADONLY**

Sets the container title to read-only. This is an attribute of the `CNRINFO` data structure's `flWindowAttr` field.

**CRA\_RECORDREADONLY**

Sets text fields in records to read-only. This is an attribute of the RECORDCORE and MINIRECORDCORE data structures' *fRecordAttr* field.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, the MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

**CFA\_FIREADONLY**

Sets column data to read-only. This is an attribute of the FIELDINFO data structure's *fData* field.

**CFA\_FITTLERADONLY**

Sets column headings to read-only. This is an attribute of the FIELDINFO data structure's *fTitle* field.

**CCS\_VERIFYPOINTERS**

A pointer verification style bit, which verifies that the application pointers are members of the container's linked list before they are used. If it is not set, the container does not verify the pointers.

**Notes:**

1. The CCS\_VERIFYPOINTERS style bit does not verify the validity of a pointer. It only verifies whether a pointer is a member of a container's linked list.
2. After your code has been developed and tested, you may want to remove the CCS\_VERIFYPOINTERS style bit in order to improve the container's performance. Otherwise, the container will attempt to verify all pointers, which will slow its response to actions that users perform.

## Container Control Selection Types

If a selection type is not specified, single selection is the default. For the tree view, single selection is the only type supported. Refer to the description of the selection types in the *SAA CUA Advanced Interface Design Reference* for more information.

**CCS\_SINGLESEL**

Single selection, which allows a user to select only one container item at a time. Each time a user selects a container item, the selection of any other container item is cancelled.

**CCS\_EXTENDSEL**

Extended selection, which allows a user to select one or more container items. A user can select one item, a range of items, or multiple ranges of items.

**CCS\_MULTIPLESEL**

Multiple selection, which allows a user to select zero or more container items.



---

## Container Control Data

See the following for information on the container control data structures:

- "CDATE" on page A-25
- "CNRDRAGINFO" on page A-28
- "CNRDRAGINIT" on page A-32
- "CNRDRAWITEMINFO" on page A-28
- "CNREDITDATA" on page A-29
- "CNRINFO" on page A-33
- "CTIME" on page A-44
- "FIELDINFO" on page A-72
- "FIELDINFOINSERT" on page A-75
- "MINIRECORDCORE" on page A-124
- "NOTIFYDELTA" on page A-130
- "NOTIFYRECORDEMPHASIS" on page A-131
- "NOTIFYRECORDENTER" on page A-132
- "NOTIFYSCROLL" on page A-133
- "OWNERBACKGROUND" on page A-135
- "QUERYRECFROMRECT" on page A-173
- "QUERYRECORDRECT" on page A-174
- "RECORDCORE" on page A-175
- "RECORDINSERT" on page A-177
- "SEARCHSTRING" on page A-184
- "TREEITEMDESC" on page A-199.

---

## Container Control Notification Messages

These messages are initiated by the container control window to notify its owner of significant events.

---

### WM\_CONTROL (in Container Controls)

For the cause of this message, see WM\_CONTROL.

#### Parameters

##### param1

##### id (USHORT)

Container control ID.

##### notifycode (USHORT)

Notify code.

The container control uses the following notification codes. For the complete description of the specified *notifycode*, see “Container Control Notification Codes” on page 22-10.

|                 |  |
|-----------------|--|
| CN_BEGINEDIT    | Container text is about to be edited.  |
| CN_COLLAPSETREE | A parent item was collapsed in the tree view.  |
| CN_CONTEXTMENU  | The container received a WM_CONTEXTMENU message.   |
| CN_DRAGAFTER    | The container received a DM_DRAGOVER message. The CN_DRAGAFTER notification code is sent only if either the CA_ORDEREDTARGETEMPH or CA_MIXEDTARGETEMPH attribute of the CNRINFO data structure is set and the current view is the name, text, or details view. |
| CN_DRAGLEAVE    | The container received a DM_DRAGLEAVE message.   |
| CN_DRAGOVER     | The container received a DM_DRAGOVER message. The CN_DRAGOVER notification code is sent only if the CA_ORDEREDTARGETEMPH attribute of the CNRINFO data structure is not set or the current view is the icon view or tree view.                                 |
| CN_DROP         | The container received a DM_DROP message.  |
| CN_DROPNOTIFY   | The container received a DM_DROPNOTIFY message.  |
| CN_DROPHELP     | The container received a DM_DROPHELP message.  |
| CN_EMPHASIS     | A container record's attributes changed.   |
| CN_ENDEDIT      | Direct editing of container text has ended.  |

|               |   |
|---------------|---|
| CN_ENTER      | The Enter key is pressed while the container window has the focus, or the select button is double-clicked while the pointer is over the container window. |
| CN_EXPANDTREE | A parent item is expanded in the tree view.   |
| CN_HELP       | The container received a WM_HELP message.   |
| CN_INITDRAG   | The drag button was pressed and the pointer was moved while the pointer was over the container control.   |
| CN_KILLFOCUS  | The container is losing the focus.  |
| CN_PICKUP     | The container received a WM_PICKUP message.   |
| CN_QUERYDELTA | Queries for more data when a user scrolls to a preset delta value.  |
| CN_REALLOCPSZ | Container text is edited. This message is sent before the CN_ENDEDIT notification code is sent.   |
| CN_SCROLL     | The container window scrolled.  |
| CN_SETFOCUS   | The container is receiving the focus.   |

**param2**

**notifyinfo (ULONG)**

Notify code information.

For the definition of this parameter, see the description of the specified *notifycode* "Container Control Notification Codes" on page 22-10 .

**Returns**

**uiReserved (ULONG)**

Reserved value, should be 0.

**Remarks**

The container control window procedure generates this message and sends it to its owner, informing the owner of this event.

**Default Processing**

For a description of the default processing, see WM\_CONTROL.

**WM\_CONTROLPOINTER (in Container Controls)**

For the cause of this message, see WM\_CONTROLPOINTER.

For a description of the parameters, see WM\_CONTROLPOINTER.

## Remarks

For the appropriate remarks, see WM\_CONTROLPOINTER.

## Default Processing

For the default processing, see WM\_CONTROLPOINTER.

---

## WM\_DRAWITEM (in Container Controls)

For the cause of this message, see WM\_DRAWITEM.

### Parameters

**param1**

**id** (USHORT)

Container control ID.

**param2**

**pOwnerItem** (POWNERITEM)

Pointer to an OWNERITEM data structure.

The following list defines the OWNERITEM data structure fields as they apply to the container control. See OWNERITEM for the default field values.

*hwnd* (HWND)

Handle of the window in which ownerdraw will occur. The following is a list of the window handles that can be specified for ownerdraw:

- The container window handle of the icon, name, text, and tree views
- The container title window handle
- The left or right window handles of the details view
- The left or right column heading windows of the details view.

*hps* (HPS)

Handle of the presentation space of the container window. For the details view that uses a split bar, the presentation space handle is either for the left or right window, depending upon the position of the column. If the details view does not have a split bar, the presentation space handle is for the left window.

*fsState* (ULONG)

Specifies emphasis flags. This state is not used by the container control because the application is responsible for drawing the emphasis states during ownerdraw.

*fsAttribute* (ULONG)

Attributes of the record as given in the *flRecordAttr* field in the RECORDCORE data structure.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of

RECORDCORE in all applicable data structures and messages. See “RECORDCORE” on page A-175 and “MINIRECORDCORE” on page A-124 for descriptions of these data structures.

*fsStateOld* (ULONG)

Previous emphasis. This state is not used by the container control because the application is responsible for drawing the emphasis states during ownerdraw.

*fsAttributeOld* (ULONG)

Previous attribute. This state is not used by the container control because the application is responsible for drawing the emphasis states during ownerdraw.

*rollItem* (RECTL)

This is the bounding rectangle into which the container item is drawn.

If the container item is an icon/text or bit-map/text pair, two WM\_DRAWITEM messages are sent to the application. The first WM\_DRAWITEM message contains the rectangle bounding the icon or bit map and the second contains the rectangle bounding the text.

If the container item contains only text, or only an icon or bit map, only one WM\_DRAWITEM message is sent. However, if the current view is the tree icon or tree text view and if the item is a parent item, the application will receive an additional WM\_DRAWITEM (in Container Controls) message. The additional message is for the icon or bit map that indicates whether the parent item is expanded or collapsed.

If the current view is the details view and the CFA\_OWNER attribute is set, the rectangle's size is equal to the width of the column and the height of the tallest field in the container item. CFA\_OWNER is an attribute of the FIELDINFO data structure's *fiData* field.

*idItem* (ULONG)

Identifies the item being drawn. It can be one of the following:

- CMA\_CNRTITLE
- CMA\_ICON
- CMA\_TEXT
- CMA\_TREEICON.

This field is not used for the details view and is set to 0.

*hlItem* (CNRDRAWITEMINFO)

Pointer to a CNRDRAWITEMINFO structure. This field is set to NULL if *idItem* is CMA\_CNRTITLE.

See “CNRDRAWITEMINFO” on page A-28 for descriptions of this structure's fields.

## Returns

**rc** (BOOL)

Item-drawn indicator.

- TRUE The owner draws the item, and so the container control does not draw it.
- FALSE If the owner does not draw the item, the owner returns this value and the container control draws the item.

## Remarks

CA\_OWNERDRAW is an attribute of the CNRINFO data structure's *flWindowAttr* field.

The container control window procedure generates this message and sends it to the owner of the container control to offer the owner the opportunity to draw that item.

## Default Processing

For a description of the default processing, see WM\_DRAWITEM.

---

## Container Control Notification Codes

The following WM\_CONTROL (in Container Controls) notification codes are sent by the container control to its owner.

---

### CN\_BEGINEDIT

The container control sends the WM\_CONTROL (in Container Controls) message with the CN\_BEGINEDIT notification code to its owner whenever container text is about to be edited.

#### Parameters

##### param1

**id** (USHORT)

Container control ID.

**CN\_BEGINEDIT** (USHORT)

Notification code.

##### param2

**pCnrEditData** (PCNREDITDATA)

Pointer to the CNREDITDATA structure.

See "CNREDITDATA" on page A-29 for definitions of this structure's fields as they apply to the CN\_BEGINEDIT notification code.

#### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Remarks

The CN\_BEGINEDIT notification code is sent when direct editing of container text begins.

**Warning:** Once your application receives the CN\_BEGINEDIT notification code, it must not send any messages to the container until it receives the CN\_ENDEDIT notification code, which indicates that direct editing of container text has ended. If any messages are sent to the container before your application receives the CN\_ENDEDIT notification code, the results of direct editing are unpredictable.

#### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_COLLAPSETREE

The container control sends the WM\_CONTROL (in Container Controls) message with the CN\_COLLAPSETREE notification code to its owner whenever the container collapses a parent item in the tree view.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_COLLAPSETREE** (USHORT)

Notification code.

#### param2

**pRecord** (RECORDCORE)

Pointer to the record that was collapsed.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.



---

## CN\_CONTEXTMENU

The container control sends the WM\_CONTROL (in Container Controls) message with the CN\_CONTEXTMENU notification code to its owner when the container receives a WM\_CONTEXTMENU message.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_CONTEXTMENU** (USHORT)

Notification code.

#### param2

**pRecord** (RECORDCORE)

Pointer to the RECORDCORE structure.

If the user is using a pointing device, this RECORDCORE structure is the structure that the pointing device pointer is over. If the pointing device pointer is over white space, this field is NULL.

If the user is using the keyboard, this RECORDCORE structure is the structure that has the selection cursor.

### Returns

**uiReserved** (ULONG)

Reserved value, should be 0.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_DRAGAFTER

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DRAGAFTER notification code to its owner whenever the container receives a DM\_DRAGOVER message. The CN\_DRAGAFTER notification code is sent only if the CA\_ORDEREDTARGETEMPHASIS or CA\_MIXEDTARGETEMPHASIS attribute of the CNRINFO data structure is set and the current view is the name, text, or details view.

## Parameters

### param1

**id** (USHORT)

Container control ID.

**CN\_DRAGAFTER** (USHORT)

Notification code.

### param2

**pCnrDragInfo** (PCNRDRAGINFO)

Pointer to a CNRDRAGINFO structure.

See "CNRDRAGINFO" on page A-28 for definitions of this structure's fields as they apply to the CN\_DRAGAFTER notification code.

## Returns

### ReturnCode

**usDrop** (USHORT)

Drop indicator.

**DOR\_DROP**

The record can be dropped. The drop will not occur unless DOR\_DROP is returned. When this response is returned, the container control applies ordered target emphasis to the target record.

**DOR\_NODROP**

The record is acceptable and the current operation is supported by the target, but the record cannot be dropped in the current location. For example, the container control returns DOR\_NODROP if the record being dragged is positioned over another record on which it cannot be dropped.

If the container returns DOR\_NODROP, the DM\_DRAGOVER message will continue to be sent to it when the user does any of the following:

- Moves the pointer
- Presses a keyboard key
- Moves the pointer out of and back into the container window.

**DOR\_NODROPOP**

The record is acceptable, but the target does not support the current operation. This response implies that the drop may be valid if the drag operation changes. For example, if the default operation is copy and the target does not support this operation, the drop may become valid if the user presses a

keyboard augmentation key to change to a different operation, such as move.

If the container returns DOR\_NODROPOP, no further DM\_DRAGOVER messages are sent until the user does any of the following:

- Presses a keyboard key
- Moves the pointer out of and back into the container window.

DOR\_NEVERDROP The record cannot be dropped. Ordered target emphasis is not drawn. If the container returns DOR\_NEVERDROP, no further DM\_DRAGOVER messages are sent until the user drags the record outside of and back into the container window.

#### **usDefaultOp (USHORT)**

Default operation.

Target-defined default operation.

DO\_COPY Operation is a copy.

DO\_DEFAULT Operation is the default drag operation. No modifier keys are pressed.

DO\_LINK Operation is a link.

DO\_MOVE Operation is a move.

DO\_UNKNOWN Operation is application-defined.

### **Remarks**

The container control draws ordered target emphasis of container records. The target emphasis provided by the container control is a black line that is drawn below the target record. Therefore, it is not necessary for the application to draw any emphasis for the container when it receives this notification code.

If the container returns anything except DOR\_DROP, the target emphasis is automatically changed to a symbol that indicates no drop is allowed. This gives the user a visual cue that a drop cannot occur. The symbol reverts to the black line when the container returns a DOR\_DROP reply.

The CN\_DRAGAFTER notification code is sent only for the details, name, and text views when the CA\_ORDEREDTARGETEMPHASIS or CA\_MIXEDTARGETEMPHASIS attribute of the CNRINFO data structure is set. If this attribute is not set, the CN\_DRAGOVER notification code is sent.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## **CN\_DRAGLEAVE**

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DRAGLEAVE notification code to its owner when the container receives a DM\_DRAGLEAVE message.

### **Parameters**

#### **param1**

**id** (USHORT)

Container control ID.

**CN\_DRAGLEAVE** (USHORT)

Notification code.

#### **param2**

**pCnrDragInfo** (PCNRDRAGINFO)

Pointer to a CNRDRAGINFO structure.

See “CNRDRAGINFO” on page A-28 for definitions of this structure’s fields as they apply to the CN\_DRAGLEAVE notification code.

### **Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Remarks**

This notification code is sent to the owner of the container control in response to a DM\_DRAGLEAVE message. It informs the owner that one of the following has occurred:

- A container record was being dragged over the container and has left the container’s boundaries.
- The drag ended when help was requested or a user pressed the Esc key while the container record was over the container.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_DRAGOVER

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DRAGOVER notification code to its owner when the container receives a DM\_DRAGOVER message. The CN\_DRAGOVER notification code is sent only if the CA\_ORDEREDTARGETEMPH attribute of the CNRINFO data structure is not set or the current view is the icon view or tree view.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_DRAGOVER** (USHORT)

Notification code.

#### param2

**pCnrDragInfo** (PCNRDRAGINFO)

Pointer to a CNRDRAGINFO structure.

See "CNRDRAGINFO" on page A-28 for definitions of this structure's fields as they apply to the CN\_DRAGOVER notification code.

### Returns

#### ReturnCode

**usDrop** (USHORT)

Drop indicator.

DOR\_DROP

The record can be dropped. When this response is returned, the container control applies target emphasis.

DOR\_NODROP

The record is acceptable and the current operation is supported by the target, but the record cannot be dropped in the current location. For example, the container control returns DOR\_DROP if the record being dragged is positioned over another record on which it cannot be dropped.

If the container returns DOR\_NODROP, the DM\_DRAGOVER message will continue to be sent to it when the user does any of the following:

- Moves the pointer
- Presses a keyboard key
- Moves the pointer out of and back into the container window.

**DOR\_NODROPOP** The record is acceptable, but the target does not support the current operation. This response implies that the drop may be valid if the drag operation changes. For example, if the default operation is copy and the target does not support this operation, the drop may become valid if the user presses a keyboard augmentation key to change to a different operation, such as move.

If the container returns **DOR\_NODROPOP**, no further **DM\_DRAGOVER** messages are sent until the user does any of the following:

- Presses a keyboard key
- Moves the pointer out of and back into the container window.

**DOR\_NEVERDROP** The record cannot be dropped. Target emphasis is not drawn. If the container returns **DOR\_NEVERDROP**, no further **DM\_DRAGOVER** messages are sent until the user drags the record outside of and back into the container window.

#### **usDefaultOp (USHORT)**

Default operation.

Target-defined default operation.

**DO\_COPY** Operation is a copy.  
**DO\_DEFAULT** Operation is the default drag operation. No modifier keys are pressed.  
**DO\_LINK** Operation is a link.  
**DO\_MOVE** Operation is a move.  
**DO\_UNKNOWN** Operation is application-defined.

#### **Remarks**

This notification code shows where direct manipulation is occurring by applying target emphasis to indicate whether an item that is being dragged over the container can be dropped. It is not necessary for the application to draw any target emphasis for the container when it receives this notification code.

If the pointer is over a container record and the item that is being dragged can be dropped on that record, the container draws a black rectangle around the target record. If the pointer is over white space and the item that is being dragged can be dropped on the white space, the container draws a black border around the edge of the client area.

If the container returns anything except **DOR\_DROP**, the target emphasis is automatically changed to a symbol that indicates no drop is allowed. This gives the user a visual cue that a drop cannot occur. The symbol reverts to the black rectangle or black border when the container returns a **DOR\_DROP** reply.

The CN\_DRAGOVER notification code is sent only for the icon and tree views, or when the CA\_ORDEREDTARGETEMPH attribute of the CNRINFO data structure is not set. If this attribute is set and the current view is the name, text, or details view, the CN\_DRAGAFTER notification code is sent.

The return parameter is reserved.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## **CN\_DROP**

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DROP notification code to its owner when the container receives a DM\_DROP message.

### **Parameters**

#### **param1**

**id** (USHORT)  
Container control ID.

**CN\_DROP** (USHORT)  
Notification code.

#### **param2**

**pCnrDragInfo** (PCNRDRAGINFO)  
Pointer to a CNRDRAGINFO structure.

See "CNRDRAGINFO" on page A-28 for definitions of this structure's fields as they apply to the CN\_DROP notification code.

### **Returns**

**ulReserved** (ULONG)  
Reserved value, should be 0.

### **Remarks**

This notification code is sent to the container's owner when dragged container records are dropped over the container window.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_DROPNOTIFY

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DROPNOTIFY notification code to its owner when a pickup set is dropped over the container.

### Parameters

#### param1

**id** (USHORT)  
Container control ID.

**CN\_DROPNOTIFY** (USHORT)  
Notification code.

#### param2

**pCnrLazyDragInfo** (PCNRLAZYDRAGINFO)  
Pointer to the CNRLAZYDRAGINFO structure.

This structure contains information about the DRAGINFO, the RECORDCORE that was dropped on, and the window handle of the target window.

### Returns

**ulReserved** (ULONG)  
Reserved value, must be 0.

### Remarks

This notification code is sent to the owner of the container when a lazy drag set is dropped over the container. (The container control receives a DM\_DROP message.)

### Default Processing

The default window procedure does not expect to receive this notification and so takes no action on it other than returning 0.

---

## CN\_DROPHELP

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_DROPHELP notification code to its owner when the container receives a DM\_DROPHELP message.

### Parameters

#### param1

**id** (USHORT)  
Container control ID.



## **CN\_DROPHELP (USHORT)**

Notification code.

### **param2**

#### **pCnrDragInfo (PCNRDRAGINFO)**

Pointer to a CNRDRAGINFO structure.

See "CNRDRAGINFO" on page A-28 for definitions of this structure's fields as they apply to the CN\_DROPHELP notification code.

### **Returns**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Remarks**

This notification code is sent to the container's owner when help for direct manipulation is requested over the container window.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## **CN\_EMPHASIS**

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_EMPHASIS notification code to its owner whenever a container record's attributes change.

### **Parameters**

#### **param1**

##### **id (USHORT)**

Container control ID.

##### **CN\_EMPHASIS (USHORT)**

Notification code.

#### **param2**

##### **pNotifyRecordEmphasis (PNOTIFYRECORDEMPHASIS)**

Pointer to the NOTIFYRECORDEMPHASIS structure.

See "NOTIFYRECORDEMPHASIS" on page A-131 for definitions of this structure's fields as they apply to the CN\_EMPHASIS notification code.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_ENDEDIT

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_ENDEDIT notification code to its owner whenever direct editing of container text has ended.

## Parameters

**param1**

**id** (USHORT)

Container control ID.

**CN\_ENDEDIT** (USHORT)

Notification code.

**param2**

**pCnrEditData** (PCNREDITDATA)

Pointer to the CNREDITDATA structure.

See "CNREDITDATA" on page A-29 for definitions of this structure's fields as they apply to the CN\_ENDEDIT notification code.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

Direct editing of container text is completed. Any changes made to the text are saved when a user presses the select button outside the window that contains the multiple-line entry (MLE) field used to edit text in a container. However, a user can end the direct editing of text without saving any changes to the text by doing any of the following:

- Pressing the Esc key
- Dragging the container item that is being edited
- Pressing the Alt key and the select button before direct editing of container text has ended
- Scrolling the container window.

The CN\_ENDEDIT notification code is sent to the application in each of these cases.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_ENTER

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_ENTER notification code to its owner when either of the following occurs:

- The Enter key is pressed while the container window has the focus
- The select button is double-clicked while the pointer is over the container window.

### Parameters

#### param1

**id** (USHORT)  
Container control ID.

**CN\_ENTER** (USHORT)  
Notification code.

#### param2

**pNotifyRecordEnter** (PNOTIFYRECORDENTER)  
Pointer to the NOTIFYRECORDENTER structure.

See "NOTIFYRECORDENTER" on page A-132 for definitions of this structure's fields as they apply to the CN\_ENTER notification code.

### Returns

**ulReserved** (ULONG)  
Reserved value, should be 0.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_EXPANDTREE

The container control sends the WM\_CONTROL (in Container Controls) message with the CN\_EXPANDTREE notification code to its owner whenever the container expands a parent item in the tree view.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_EXPANDTREE** (USHORT)

Notification code.

#### param2

**pRecord** (RECORDCORE)

Pointer to the record that was expanded.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_HELP

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_HELP notification code to its owner whenever the container receives a WM\_HELP message.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

## **CN\_HELP (USHORT)**

Notification code.

### **param2**

#### **pRecord (PRECORDCORE)**

Pointer to the record that has the selection cursor.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### **Returns**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Remarks**

This notification code is sent to the container's owner when help is requested for a container item.

### **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## **CN\_INITDRAG**

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_INITDRAG notification code to its owner when the drag button is pressed and the pointer is moved while the pointer is over the container control.

### **Parameters**

#### **param1**

##### **id (USHORT)**

Container control ID.

##### **CN\_INITDRAG (USHORT)**

Notification code.

#### **param2**

##### **pCnrDragInit (PCNRDRAGINIT)**

Pointer to the CNRDRAGINIT structure.

See "CNRDRAGINIT" on page A-32 for descriptions of this structure's fields as they apply to the CN\_INITDRAG notification code.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

This notification code is sent to the container's owner when the drag button is pressed and the pointer is moved while the pointer is over the container control.

## Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_KILLFOCUS

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_KILLFOCUS notification code to its owner whenever the container is losing the focus.

## Parameters

**param1**

**id** (USHORT)

Container control ID.

**CN\_KILLFOCUS** (USHORT)

Notification code.

**param2**

**hwndCnr** (HWND)

Container control handle.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_PICKUP

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_PICKUP notification code to its owner when a pickup and drop operation is initiated over a container.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_PICKUP** (USHORT)

Notification code.

#### param2

**pCnrDragInit** (PCNRDRAGINIT)

Pointer to the CNRDRAGINIT structure containing direct-manipulation information initiated in a container.

The CNRDRAGINIT structure is the same as the one used for standard drag notifications.

### Returns

#### returns

**ulReserved** (ULONG)

Reserved value, must be 0.

### Remarks

This notification code is sent to the owner of the container when a lazy drag operation is commenced over a container. (The container control receives a WM\_PICKUP message.)

The CN\_PICKUP message handler determines if the mouse is over an object or in white space of the client window.

If a pickup object is not selected, only that pickup object is added to the lazy drag set. If the pickup object is selected, all selected items in the container are added to the lazy drag set. The shell sets the CRA\_PICKED attributes for all objects that are picked.

### Default Processing

The default message procedure sets *ulReserved* to 0.

---

## CN\_QUERYDELTA

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_QUERYDELTA notification code to its owner to query for more data when a user scrolls to a preset delta value.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_QUERYDELTA** (USHORT)

Notification code.

#### param2

**pNotifyDelta** (PNOTIFYDELTA)

Pointer to the NOTIFYDELTA structure.

See "NOTIFYDELTA" on page A-130 for definitions of this structure's fields as they apply to the CN\_QUERYDELTA notification code.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The delta value is specified by the *cDelta* field of the CNRINFO data structure and is set with the CMA\_DELTA attribute of the CM\_SETCNRINFO message. If the value of the *cDelta* field is greater than 0 and a user scrolls to the threshold record, the container control sends a CN\_QUERYDELTA notification code to the application. The application can then insert more records into the container. It may be necessary for the application to remove some records before inserting records.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.



---

## CN\_REALLOCPSZ

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_REALLOCPSZ notification code to its owner whenever container text is edited. It is sent before the CN\_ENDEDIT notification code is sent.

### Parameters

#### param1

**id** (USHORT)

Container control ID.

**CN\_REALLOCPSZ** (USHORT)

Notification code.

#### param2

**pCnrEditData** (PCNREDITDATA)

Pointer to the CNREDITDATA structure.

See "CNREDITDATA" on page A-29 for definitions of this structure's fields as they apply to the CN\_REALLOCPSZ notification code.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** The application has sufficient memory for the new text string.

**FALSE** The application has insufficient memory for the new text string or does not want the string to be copied.

### Remarks

The CN\_REALLOCPSZ notification code is sent after direct editing of container text is complete. It notifies the application that the container is about to copy the changed text to the application's text string. This allows the application to ensure that the correct amount of memory is allocated to accommodate the change.

If TRUE is returned by the application, the container control copies the new text to the application's text string. However, if the application returns FALSE, changed text is disregarded. **Warning:** Once your application receives the CN\_REALLOCPSZ notification code, it must not send any messages to the container until it receives the CN\_ENDEDIT notification code, which indicates that direct editing of container text has ended. If any messages are sent to the container before your application receives the CN\_ENDEDIT notification code, the results of direct editing are unpredictable.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return FALSE.

---

## CN\_SCROLL

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_SCROLL notification code to its owner whenever the container window scrolls.

### Parameters

#### param1

**id** (USHORT)  
Container control ID.

**CN\_SCROLL** (USHORT)  
Notification code.

#### param2

**pNotifyScroll** (PNOTIFYSCROLL)  
Pointer to the NOTIFYSCROLL structure.

See "NOTIFYSCROLL" on page A-133 for definitions of this structure's fields as they apply to the CN\_SCROLL notification code.

### Returns

**rc** (ULONG)  
Reserved value, should be 0.

### Default Processing

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## CN\_SETFOCUS

The container control sends a WM\_CONTROL (in Container Controls) message with the CN\_SETFOCUS notification code to its owner whenever the container receives the focus.

### Parameters

#### param1

**id** (USHORT)  
Container control ID.

**CN\_SETFOCUS** (USHORT)  
Notification code.

## **param2**

**hwndCnr** (HWND)

Container control handle.

## **Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

## **Default Processing**

The default window procedure does not expect to receive this notification code and therefore takes no action on it other than to return 0.

---

## Container Control Window Messages

This section describes the container control window procedure actions on receiving the following messages.

---

### CM\_ALLOCDETAILFIELDINFO

This message allocates memory for one or more FIELDINFO structures.

#### Parameters

##### param1

##### nFieldInfo (USHORT)

Number of FIELDINFO structures to be allocated.

The value of this parameter must be greater than 0.

##### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

#### Returns

##### pFieldInfo (PFIELDINFO)

Pointer or error.

0 Reserved value, 0. The WinGetLastError function may return the following errors:

- PMERR\_INSUFFICIENT\_MEMORY
- PMERR\_INVALID\_PARAMETERS.

Other If the *nFieldInfo* parameter has a value of 1, a pointer to a FIELDINFO data structure is returned.

A pointer to the first FIELDINFO structure in a linked list of FIELDINFO structures is returned if the *nFieldInfo* parameter has a value greater than 1. The pointer to the next FIELDINFO structure is set in each *pNextFieldInfo* field of the FIELDINFO data structure. The last pointer is set to NULL.

## Remarks

The container control requires that the application use the CM\_ALLOCDETAILFIELDINFO message to allocate memory for any FIELDINFO structures that are used.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_ALLOCRECORD

This message allocates memory for one or more RECORDCORE structures.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

## Parameters

### param1

#### cbRecordData (ULONG)

Bytes of additional memory.

The number of bytes of additional memory that you want to reserve for your application's private use. This parameter must have a value between 0 and 64,000. If the value is 0, no additional memory is allocated, but a RECORDCORE data structure is allocated.

### param2

#### nRecords (USHORT)

Number of records.

The number of container records to be allocated. This parameter must have a value greater than 0.

## Returns

### pRecord (RECORDCORE)

Returns a pointer or an error.

NULL Allocation failed. The WinGetLastError function may return the following errors:

- PMERR\_INSUFFICIENT\_MEMORY
- PMERR\_INVALID\_PARAMETERS.

Other If the *nRecords* parameter has a value of 1, a pointer to a RECORDCORE structure is returned.

If the *nRecords* parameter has a value greater than 1, a pointer to the first RECORDCORE structure in the linked list of records is returned. The pointer to the next container record is set in the *preccNextRecord* field in each RECORDCORE data structure. The last pointer is set to NULL.

## Remarks

The container control requires that the application use the CM\_ALLOCORECORD message to allocate memory for container records.

When a record is allocated, the *cb* field of the record will be initialized with the size of the record structure type currently in use, either RECORDCORE or MINIRECORDCORE. If the CCS\_MINIRECORDCORE style bit is not specified, the record is allocated according to the size of the RECORDCORE data structure. However, if the CCS\_MINIRECORDCORE style bit is specified, the record is allocated according to the size of the MINIRECORDCORE data structure. This size should not be modified by the application.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_ARRANGE

This message arranges the container records in the icon view of the container control.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Icon/text or bit-map/text pairs were successfully arranged.

FALSE An error occurred.

### Remarks

The container items fill the topmost row until the width of the client area is reached. The container items then wrap to form another row immediately below the filled row. This process is repeated until all of the container items are positioned in rows. Default spacing is implemented according to the guidelines for the CUA user interface. A vertical scroll bar is enabled, if necessary.

Before the relocation of the container items, the origin of the client area rectangle is reset to coincide with the origin of the container's workspace. Arranging the container items does not affect the record attributes.

If the CCS\_AUTOPOSITION style bit is set, you do not need to send the CM\_ARRANGE message, since this style bit causes the container control to arrange the container items for the application.

If the current view is not the icon view, no visible change occurs until the current view is switched to the icon view. For example, if the name view is the current view and the CM\_ARRANGE message is sent, the display does not change.

The container updates the *ptIcon* field of the RECORDCORE structure with the new coordinates.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_CLOSEEDIT

This message closes the window that contains the multiple-line entry (MLE) field used to edit container text directly.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** The direct editing of container item text was successfully ended.

**FALSE** The direct editing of container item text was not successfully ended. The WinGetLastError function may return the following error:

PMERR\_INSUFFICIENT\_MEMORY.

### Remarks

The application sends this message to the container control to end the direct editing of container text. The application can assign this message to a key or key combination, a menu choice, or both so that the user can end the direct editing of container text from the keyboard.

When the container control receives this message, it sends the CN\_REALLOCPSZ and CN\_ENDEDIT notification codes to the application.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.



---

## CM\_COLLAPSETREE

This message causes one parent item in the tree view to be collapsed.

### Parameters

#### param1

##### **pRecord** (RECORDCORE)

Pointer to the RECORDCORE structure that is to be collapsed.

If this is NULL, all expanded parent items are collapsed.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

TRUE The item was successfully collapsed.

FALSE An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_ERASERECORD

This message erases the source record from the current view when a move occurs as a result of direct manipulation.

### Parameters

**param1**

**pRecord** (PRECORDCORE)

Pointer to the container record that is to be erased from the current view.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE The record was successfully erased.

FALSE The record was not erased. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_INSUFFICIENT\_MEMORY.

### Remarks

The container record is not removed and memory is not freed; only the visual appearance is changed. The visibility flag associated with the container record is not changed.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_EXPANDTREE

This message causes one parent item in the tree view to be expanded.

### Parameters

#### param1

##### pRecord (RECORDCORE)

Pointer to the RECORDCORE structure that is to be expanded.

If this is NULL, all collapsed parent items are expanded.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

TRUE The item was successfully expanded.

FALSE An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_FILTER

This message filters the contents of a container so that a subset of the container items is viewable.

### Parameters

**param1**

**pfnFilter** (PFN)

Pointer to an application-supplied filter function.

**param2**

**pStorage** (PVOID)

Application use.

Available for application use.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** A subset was successfully created.

**FALSE** An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_NO\_FILTERED\_ITEMS
- PMERR\_INSUFFICIENT\_MEMORY.

### Remarks

Filtering is enabled by setting the CRA\_FILTERED attribute of container records that are to be excluded from the viewable subset.

The *pfnFilter* parameter points to an application-provided function that determines whether a record is to be included in the viewable subset. The *pfnFilter* parameter must be declared as:

```
BOOL PFN pfnFilter (RECORDCORE p, PVOID pStorage);
```

where **p** points to a RECORDCORE structure that describes the container record to be tested. The *pfnFilter* parameter returns TRUE if the record is to be included in the viewable subset, or FALSE if it is to be excluded. The container sets the CRA\_FILTERED attribute for the record based on the return from the *pfnFilter* parameter.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

If the `CRA_FILTERED` attribute is set for the record, the record is not visible. If the `CCS_AUTOPOSITION` style bit is set and the container is showing the icon view, the container records are arranged when a record is filtered out.

The `CM_FILTER` message supports only one level of filtering.

It is the application's responsibility to provide a National Language Support-enabled (NLS-enabled) function for the *pfnFilter* parameter.

If the *pfnFilter* parameter value is `NULL`, a container is returned to an unfiltered state. If functions such as inserting a record into a container, arranging the records, or sorting the records are performed on a container whose records have been filtered, the effect of these functions remains if the container records are later unfiltered.

All messages act on the entire container. For example, a record that is filtered and is removed from the container will be removed from the container entirely; it is not present in the container when the container records are unfiltered.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `FALSE`.

---

## CM\_FREEDTAILFIELDINFO

This message frees the memory associated with one or more `FIELDINFO` structures.

### Parameters

**param1**

**pFieldInfoArray** (PVOID)

Pointer to an array of pointers to `FIELDINFO` structures that are to be freed.

**param2**

**cNumFieldInfo** (USHORT)

Number of structures.

Number of `FIELDINFO` structures to be freed.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Memory associated with a specified FIELDINFO structure or structures in the container was freed.

**FALSE** Associated memory was not freed. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_MEMORY\_DEALLOCATION\_ERR
- PMERR\_FI\_CURRENTLY\_INSERTED.

## Remarks

It is the application's responsibility to free all application-allocated memory associated with the structures, such as user data.

If a specified FIELDINFO structure is currently inserted into the container, the structure is not freed and the PMERR\_FI\_CURRENTLY\_INSERTED error is set. FIELDINFO structures must be removed with the CM\_REMOVEDTAILFIELDINFO message before the CM\_FREEDETAILFIELDINFO message is used.

If the number of pointers to FIELDINFO structures in the array exceeds the count of structures to be freed, only the number of structures in the *cNumFieldInfo* parameter is freed. If either the *pFieldInfoArray* or the *cNumFieldInfo* parameter is invalid, the PMERR\_INVALID\_PARAMETERS error is set and no FIELDINFO structures are freed.

If the PMERR\_MEMORY\_DEALLOCATION\_ERR error occurs, any further processing is unreliable.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_FREERECORD

This message frees the memory associated with one or more RECORDCORE structures.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Parameters

**param1**

**pRecordArray** (PVOID)

Pointer to an array of pointers to RECORDCORE structures that are to be freed.

**param2**

**cNumRecord** (USHORT)

Number of records.

Number of container records to be freed.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Memory associated with a record or records in the container was freed.

FALSE Associated memory was not freed. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_MEMORY\_DEALLOCATION\_ERR
- PMERR\_RECORD\_CURRENTLY\_INSERTED.

### Remarks

It is the application's responsibility to free all application-allocated memory associated with the container records, such as text strings.

If a specified record is currently inserted into the container, the record is not freed and the PMERR\_RECORD\_CURRENTLY\_INSERTED error is set. Container records must be removed with the CM\_REMOVERECORD message before the CM\_FREERECORD message is used.

If the number of pointers to container records in the array exceeds the count of records to be freed, only the number of records in the *cNumRecord* parameter is freed. If either the *pRecordArray* or the *cNumRecord* parameter is invalid, the PMERR\_INVALID\_PARAMETERS error is set and no container records are freed.

If the `PMERR_MEMORY_DEALLOCATION_ERR` error occurs, any further processing is unreliable.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `FALSE`.

---

## CM\_HORZSCROLLSPLITWINDOW

This message scrolls a split window in the split details view.

### Parameters

#### param1

##### **usWindow** (USHORT)

Window indicator.

`CMA_LEFT`     The left split window is scrolled.

`CMA_RIGHT`    The right split window is scrolled.

#### param2

##### **IScrollInc** (LONG)

Amount to scroll.

Amount (in pixels) by which to scroll the window.

### Returns

#### **rc** (BOOL)

Success indicator.

`TRUE`     Successful completion

`FALSE`    An error occurred. The `WinGetLastError` function may return the following error:

`PMERR_INVALID_PARAMETERS`.

### Remarks

The *IScrollInc* parameter indicates a change in position. If the *IScrollInc* parameter value is greater than 0, the window specified in the *usWindow* parameter is scrolled to the right by the number of pixels specified in the *IScrollInc* parameter. If the value of the *IScrollInc* parameter is less than 0, the window specified in the *usWindow* parameter is scrolled to the left by the number of pixels specified in the *IScrollInc* parameter. This message is used to scroll either the left or right split window by an absolute amount.

The columns that are to appear in each split window are determined at the time the split window is created. Thereafter, columns in the left split window cannot be seen in the right split window, and vice versa.



## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_INSERTDETAILFIELDINFO

This message inserts one or more FIELDINFO structures into a container control.

### Parameters

**param1**

**pFieldInfo** (PFIELDINFO)

Pointer to the FIELDINFO structure or structures to insert.

**param2**

**pFieldInfoInsert** (PFIELDINFOINSERT)

Pointer to the FIELDINFOINSERT data structure.

See "FIELDINFOINSERT" on page A-75 for the descriptions of this structure's fields as they apply to the CM\_INSERTDETAILFIELDINFO message.

### Returns

**cFields** (USHORT)

Number of structures.

0 The FIELDINFO structure or structures were not inserted. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_INSUFFICIENT\_MEMORY
- PMERR\_FI\_CURRENTLY\_INSERTED.

Other The number of FIELDINFO structures in the container.

### Remarks

The *pFieldInfoInsert* parameter is used to insert FIELDINFO structures into the container. The *pFieldInfoOrder* field of the FIELDINFOINSERT data structure is used to place FIELDINFO structures into the container in order, relative to the other structures. Specifying the CMA\_FIRST attribute places the FIELDINFO structure at the front of the list of structures. If the CMA\_END attribute is specified, the FIELDINFO structure is placed at the end of the list of structures. Otherwise, if the value of the *pFieldInfoOrder* field is a pointer to a FIELDINFO structure, the structure being inserted is placed after this structure.

If the value of the *cFieldInfoInsert* field of the FIELDINFOINSERT data structure is greater than 1, a linked list of FIELDINFO structures is inserted in the order specified by the *pFieldInfoOrder* field. Here, the *pFieldInfo* parameter points to the first of a linked list of FIELDINFO structures. This list of structures is linked together as they were when the FIELDINFO structures were allocated.

If one FIELDINFO structure is to be inserted, the *cFieldInfoInsert* field has a value of 1 and the *pFieldInfo* parameter points to the FIELDINFO structure to be inserted.

After the FIELDINFO structures have been inserted, if the *fnvalidateFieldInfo* field of the FIELDINFOINSERT data structure is FALSE, the CM\_INVALIDATEDDETAILFIELDINFO message must be sent to update the display with the inserted structures.

If the CCS\_VERIFYPOINTERS style bit is set and the *pFieldInfo* parameter contains a pointer to a FIELDINFO structure that is currently inserted, the PMERR\_FI\_CURRENTLY\_INSERTED error is set and no FIELDINFO structures are inserted.

### **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## **CM\_INSERTRECORD**

This message inserts one or more RECORDCORE structures into a container control.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### **Parameters**

**param1**

**pRecord** (PRECORDCORE)

Pointer to the RECORDCORE structure or structures to insert.

**param2**

**pRecordInsert** (RECORDINSERT)

Pointer to the RECORDINSERT data structure.

See "RECORDINSERT" on page A-177 for definitions of this structure's fields as they apply to the CM\_INSERTRECORD message.

## Returns

### cRecords (ULONG)

Number of structures.

- 0        The RECORDCORE structure was not inserted. The WinGetLastError function may return the following errors:
- PMERR\_INVALID\_PARAMETERS
  - PMERR\_INSUFFICIENT\_MEMORY
  - PMERR\_RECORD\_CURRENTLY\_INSERTED.
- Other    The number of RECORDCORE structures in the container.

## Remarks

The *pRecordInsert* parameter is used to insert RECORDCORE structures into the container. The *pRecordOrder* and *pRecordParent* fields of the RECORDINSERT data structure are used to place each record into the container in order, relative to the other records. If the CMA\_FIRST or CMA\_END attributes are specified, records are inserted before the first child or after the last child of the record specified in the *pRecordParent* field. If the value of the *pRecordParent* field is NULL, the record or records are inserted before the first record or after the last record, respectively, at the root level. Otherwise, if the value of the *pRecordOrder* field is a pointer to a record, the record or records to be inserted are placed after this record.

A z-ordering of the records is maintained by the container control. The *zOrder* field of the RECORDINSERT data structure is used to specify the record's z-order in the container, relative to the other records. The CMA\_TOP attribute is used to place the record at the end of the z-order list, while the CMA\_BOTTOM attribute places the record at the beginning of the z-order list. Z-ordering is used for the icon view only.

If the value of the *cRecordsInsert* field of the RECORDINSERT data structure is greater than 1, a linked list of RECORDCORE structures is inserted in the order specified by the *pRecordOrder*, *pRecordParent*, and *zOrder* fields. Here, the *pRecord* parameter points to the first RECORDCORE structure of a linked list of structures.

If one RECORDCORE structure is to be inserted, the *cRecordsInsert* field has a value of 1 and the *pRecord* parameter points to the RECORDCORE structure to be inserted.

When containers display the icon view, the coordinates specified by the RECORDCORE structure's *ptIcon* field are used to position inserted container records in the container's workspace. If the coordinates are not specified and the CCS\_AUTOPOSITION style bit is not set, all of the inserted container records are positioned at (0,0) and a CM\_ARRANGE message must be sent to position them elsewhere. If the CCS\_AUTOPOSITION style bit is set, the container records are positioned without the CM\_ARRANGE message being sent.

After the container records have been inserted:

- If the *flInvalidateRecord* field of the RECORDINSERT data structure is FALSE, the CM\_INVALIDATERECORD message must be sent to update the display with the inserted records. If the current view is the icon view and either the

CCS\_AUTOPOSITION style bit is set or the *flinvalidateRecord* field is TRUE, the view is updated without the CM\_INVALIDATERECORD message being sent.

- The *preccNextRecord*, *flRecordAttr*, and *ptllcon* fields of the external RECORDCORE structure are not updated as changes occur within the container. However, if records are shared among multiple containers, the *flRecordAttr* and *ptllcon* fields are modified internally. Refer to the *OS/2 2.00 Programming Guide* for more information about the modification of these fields.

If the CCS\_VERIFYPOINTERS style bit is set and the *pRecord* parameter contains a pointer to a RECORDCORE structure that is currently inserted, the PMERR\_RECORD\_CURRENTLY\_INSERTED error is set and no RECORDCORE structures are inserted.

If the RECORDCORE structures are sorted on insertion, the *pRecordOrder* and *zOrder* fields are ignored.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## CM\_INSERTRECORDARRAY

This message inserts one or more RECORDCORE structures into a container control.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container control is created, then MINIRECORDCORE should be used instead of RECORDCORE, and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Parameters

**param1**

**pRecordArray** (PVOID)

Pointer to an array of pointers to RECORDCORE structures that are to be inserted into the container.

**param2**

**pRecordInsert** (PRECORDINSERT)

Pointer to the RECORDINSERT structure.

## Returns

### **cRecords** (ULONG)

Number of RECORDCORE structures in the root level of the container.

0 No RECORDCORE structures were inserted.

The WinGetLastError function may return the following errors:

PMERR\_INVALID\_PARAMETERS  
PMERR\_INSUFFICIENT\_MEMORY  
PMERR\_RECORD\_CURRENTLY\_INSERTED

Other The number of RECORDCORE structures in the container.

## Remarks

The *pRecordInsert* parameter is used to insert RECORDCORE structures into the container. The *pRecordOrder* and *pRecordParent* fields of the RECORDINSERT data structure are used to place each record into the container in order, relative to the other records. If the CMA\_FIRST or CMA\_END attributes are specified, records are inserted before the first child or after the last child of the record specified in the *pRecordParent* field. If the value of the *pRecordParent* field is NULL, the record or records are inserted before the first record or after the last record, respectively, at the root level. Otherwise, if the value of the *pRecordOrder* field is a pointer to a record, the record or records to be inserted are placed after this record.

A z-ordering of the records is maintained by the container control. The *zOrder* field of the RECORDINSERT data structure is used to specify the record's z-order in the container, relative to the other records. The CMA\_TOP attribute is used to place the record at the end of the z-order list, while the CMA\_BOTTOM attribute places the record at the beginning of the z-order list. Z-ordering is used for the icon view only.

The *cRecords* parameter always specifies an array of pointers to RECORDCORE structures to be inserted into the container. The number of pointers contained in the array must equal the value specified in the *cRecordsInsert* field of the RECORDINSERT structure.

When containers display the icon view, the coordinates specified by the RECORDCORE structure's *ptIcon* field are used to position inserted container records in the container's workspace. If the coordinates are not specified and the CCS\_AUTOPOSITION style bit is not set, all of the inserted container records are positioned at (0,0) and a CM\_ARRANGE message must be sent to position them elsewhere. If the CCS\_AUTOPOSITION style bit is set, the container records are positioned without the CM\_ARRANGE message being sent.

After the container records have been inserted:

- If the *flinvalidateRecord* field of the RECORDINSERT data structure is FALSE, the CM\_INVALIDATERECORD message must be sent to update the display with the inserted records. If the current view is the icon view and either the CCS\_AUTOPOSITION style bit is set or the *flinvalidateRecord* field is TRUE, the view is updated without the CM\_INVALIDATERECORD message being sent.
- The *preccNextRecord*, *flRecordAttr*, and *ptilcon* fields of the external RECORDCORE structure are not updated as changes occur within the container. However, if records are shared among multiple containers, the *flRecordAttr* and *ptilcon* fields are modified internally.

If the CCS\_VERIFYPOINTERS style bit is set and the *pRecordArray* parameter contains a pointer to a RECORDCORE structure that is currently inserted, the PMERR\_RECORD\_CURRENTLY\_INSERTED error is set and no RECORDCORE structures are inserted.

If the RECORDCORE structures are sorted on insertion, the *pRecordOrder* and *zOrder* fields are ignored.

### Default Processing

The default window procedure does not expect to receive this message and, therefore, takes no action on it other than to return FALSE.

---

## CM\_INVALIDATEDDETAILFIELDINFO

This message notifies the container control that any or all FIELDINFO structures are not valid and that the view must be refreshed.

### Parameters

**param1**

**ulReserved (ULONG)**

Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### Returns

**rc (BOOL)**

Success indicator.

TRUE FIELDINFO structures were successfully refreshed.

FALSE FIELDINFO structures were not successfully refreshed.

## Remarks

If any or all FIELDINFO structures are changed, removed, or inserted, the CM\_INVALIDATEDDETAILFIELDINFO message must be sent. Since each FIELDINFO structure potentially affects every record in the container, the entire view is refreshed, even if only one FIELDINFO structure has changed.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_INVALIDATERECORD

This message notifies the container control that a RECORDCORE structure or structures are not valid and must be refreshed.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

## Parameters

### param1

#### **pRecordArray** (PVOID)

Pointer to an array of pointers to RECORDCORE structures that are to be refreshed.

### param2

#### **cNumRecord** (USHORT)

Number of container records to be refreshed.

If the *cNumRecord* parameter has a value of 0, all of the records in the container are refreshed and the *pRecordArray* parameter is ignored.

#### **fInvalidateRecord** (USHORT)

Flags used to optimize container record invalidation.

The CMA\_REPOSITION, CMA\_NOREPOSITION, and CMA\_TEXTCHANGED attributes are mutually exclusive. However, any of them can be combined with the CMA\_ERASE attribute by using a logical OR operator (|).

#### **CMA\_ERASE**

Flag used when the icon view is displayed to minimize painting of a container record's background when it has changed. If specified, the background is erased when the display is refreshed. The default is to not erase the background when the display is refreshed.

**CMA\_REPOSITION** Flag used to reposition all container records. This flag must be used if container records are inserted or removed, or if many changes have occurred. If a container record is inserted, the *pRecordArray* parameter points to the inserted record. If a container record is removed, the *pRecordArray* parameter points to the record that precedes the removed one. If several container records have changed, an array of container record pointers must be used. The container determines the first record to be invalidated. This is the default.

**CMA\_NOREPOSITION** Flag used to indicate that container records do not need to be repositioned. The container draws the record or records pointed to in the *pRecordArray* parameter. The container does not do any validation; therefore it is the application's responsibility to make sure repositioning is not needed or changing the longest text line is not necessary.

**CMA\_TEXTCHANGED** Flag used if text has changed and you do not know whether repositioning is needed. The container determines whether the longest line or the height of the record has changed. If so, the container repositions and redraws the necessary visible container records.

It may be necessary to reposition the container records if the number of lines of text has changed. **Warning:** The application must send a `CM_INVALIDATERECORD` message if text changes. Otherwise, any further processing is unreliable.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Records were successfully refreshed.

**FALSE** An error occurred. The `WinGetLastError` function may return the following errors:

- `PMERR_INVALID_PARAMETERS`
- `PMERR_INSUFFICIENT_MEMORY`.

## Remarks

If the number of pointers to container records in the array exceeds the count of records to be refreshed, only the number of records specified in the *cNumRecord* parameter is refreshed. If the `CCS_VERIFYPOINTERS` style bit is set and the *pRecordArray* parameter contains pointers to a `RECORDCORE` structure or structures that do not exist, the `PMERR_INVALID_PARAMETERS` error is set and nothing is refreshed.



## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_MOVETREE

This message is used to move a record to a new parent in the container control.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container control is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Parameters

**param1**

**pTreeMove** (PTREEMOVE)

Pointer to a TREEMOVE structure.

See TREEMOVE for definitions of this structure's fields as they apply to the CM\_MOVETREE message.

**param2**

**Reserved** (ULONG)

Reserved value, must be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Record and associated subtrees were moved successfully.  
FALSE Error occurred, and tree structure remains unchanged.

### Remarks

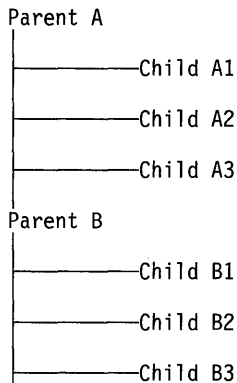
This message is used to change the parent of a record in the container control. The fields of the TREEMOVE structure describe the record to be moved, the record to become its new parent, and where to insert the record relative to other records with the same parent.

If the *preccNewParent* field of the TREEMOVE structure is NULL, the record being moved is moved to the root level; otherwise, it is moved to *preccNewParent*. The *pRecordOrder* field of the TREEMOVE structure determines where the record being moved is placed relative to other records with the same parent (the one specified by *preccNewParent*). If *flMoveSiblings* of the TREEMOVE structure is TRUE, all siblings that follow the record being moved (*preccMove*) are moved to the new parent as well. Siblings that precede *preccMove* are not moved regardless of the value of the *flMoveSiblings* field. For normal Tree Move operations, the *flMoveSiblings* field of the TREEMOVE structure should be set to FALSE.

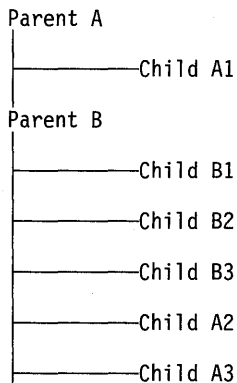
WinGetLastError returns PMERR\_INVALID\_PARAMETERS if any of the following illegal combinations are used:

- *flMoveSiblings* is either the first or last root level record in the container, and the *flMoveSiblings* flag is TRUE.
- *preccMove* is a root level record, and *preccNewParent* is currently one of its children.
- *pRecordOrder* is a pointer to a RECORDCORE structure (not CMA\_FIRST or CMA\_LAST) tha does not exist in the list of children of the new parent.
- *preccNewParent* is NULL, and *pRecordOrder* is not a root level record.

For example, the following tree contains two parents, each with three children:



If *preccMove* is Child A2, *preccNewParent* is Parent B, *pRecordOrder* = CMA\_LAST and *flMoveSiblings* = TRUE, after the Tree Move operation, the new tree structure is as follows:



### Default Processing

The default window procedure does not expect to receive this message and, therefore, takes no action on it other than to return FALSE.

---

## CM\_OPENEDIT

This message opens the window that contains the multiple-line entry (MLE) field used to edit container text directly.

### Parameters

#### param1

##### **pCnrEditData** (PCNREDITDATA)

Pointer to the CNREDITDATA structure.

See "CNREDITDATA" on page A-29 for definitions of this structure's fields as they apply to the CM\_OPENEDIT message.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### rc (BOOL)

Success indicator.

- TRUE Direct editing of container text was successfully started.
- FALSE Direct editing of container text was not successfully started. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Remarks

The application sends this message to the container control to start the direct editing of container text. The application can assign this message to a key or key combination, a menu choice, or both so that the user can start editing container text directly from the keyboard.

When the container control receives this message, it sends the CN\_BEGINEDIT notification code to the application.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_PAINTBACKGROUND

This message informs an application whenever a container's background is painted if the CA\_OWNERPAINTBACKGROUND attribute of the CNRINFO data structure is specified.

### Parameters

#### param1

##### **pOwnerBackground** (POWNERBACKGROUND)

Pointer to the OWNERBACKGROUND structure.

See "OWNERBACKGROUND" on page A-135 for definitions of this structure's fields as they apply to the CM\_PAINTBACKGROUND message.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **rc** (BOOL)

Process indicator.

TRUE The application processed the CM\_PAINTBACKGROUND message.

FALSE The application did not process the CM\_PAINTBACKGROUND message.

### Remarks

The CM\_PAINTBACKGROUND message is provided so that an application can subclass the container control and paint its own background. If the application does not subclass the container control or subclasses the container control and returns FALSE, the container uses the system window color, which is specified by SYSCLR\_WINDOW. This color can be changed by using the PP\_BACKGROUND\_COLOR or PP\_BACKGROUND\_COLOR\_INDEX presentation parameter of the WM\_PRESPARAMCHANGED (in Container Controls) message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_QUERYCNRINFO

This message returns the container's CNRINFO structure.

### Parameters

#### param1

##### **pCnrInfo** (PCNRINFO)

Pointer to a buffer into which the CNRINFO structure is copied.

#### param2

##### **cbBuffer** (USHORT)

Number of bytes.

Maximum number of bytes to copy.

### Returns

#### **cbBytes** (USHORT)

Success indicator.

0 Container data was not successfully returned. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

Other Actual number of bytes copied.

### Remarks

The number of bytes specified in the *cbBuffer* parameter is returned in the buffer addressed by the *pCnrInfo* parameter.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## CM\_QUERYDETAILFIELDINFO

This message returns a pointer to the requested FIELDINFO structure.

### Parameters

#### param1

##### **pfldinfoBase** (PFIELDINFO)

Pointer to the FIELDINFO structure used to search for the next or previous column.

If the CMA\_FIRST or CMA\_LAST attribute is specified, this is ignored.

## param2

### cmd (USHORT)

Command that indicates which FIELDINFO structure to retrieve.

|           |                                   |
|-----------|-----------------------------------|
| CMA_FIRST | First column in the container.    |
| CMA_LAST  | Last column in the container.     |
| CMA_NEXT  | Next column in the container.     |
| CMA_PREV  | Previous column in the container. |

## Returns

### pFieldInfo (PFIELDINFO)

Pointer to the FIELDINFO structure for which data was requested.

NULL No FIELDINFO structures to retrieve.

-1 The data from the FIELDINFO structure was not returned. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

Other Pointer to the FIELDINFO structure for which data was requested.

## Remarks

If the *cmd* parameter has the value of the CMA\_FIRST or CMA\_LAST attribute, the *pFldInfoBase* parameter is ignored and the first or last column data, respectively, is returned. If the CMA\_NEXT or the CMA\_PREV attribute is set in the *cmd* parameter, the column data next to or before the column pointed to by the *pFieldInfo* parameter is returned.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_QUERYDRAGIMAGE

This message returns a handle to the icon or bit map for the record in the current view.

## Parameters

### param1

### pRecord (RECORDCORE)

Pointer to the RECORDCORE structure that is to be queried for the image.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

## param2

### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

### hImage (LHANDLE)

Image handle.

NULLHANDLE If no image is defined, NULLHANDLE is returned.

Other Handle of an icon or bit map.

- If the CA\_DRAWICON attribute and the CV\_MINI style bit are specified, the RECORDCORE structure's *hptrMiniIcon* field is returned.
- If the CA\_DRAWICON attribute is specified without the CV\_MINI style bit, the RECORDCORE structure's *hptrIcon* field is returned.
- If the CA\_DRAWBITMAP attribute and the CV\_MINI style bit are specified, the RECORDCORE structure's *hbmMiniBitmap* field is returned.
- If the CA\_DRAWBITMAP attribute is specified without the CV\_MINI style bit, the RECORDCORE structure's *hbmBitmap* field is returned.

## Remarks

If the CCS\_MINIRECORDCORE style bit is specified, this function will always return the MINIRECORDCORE structure's *hptrIcon* field.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULLHANDLE.

---

## CM\_QUERYRECORD

This message returns a pointer to the requested RECORDCORE structure.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Parameters

#### param1

##### pRecord (PRECORDCORE)

Pointer to the RECORDCORE structure used to search for the next or previous container record.

If the CMA\_FIRST or CMA\_LAST attribute is specified, this is ignored.

#### param2

##### cmd (USHORT)

Command that indicates which container record to retrieve:

|                |   |
|----------------|---|
| CMA_FIRST      | First record in the container.                                    |
| CMA_FIRSTCHILD | First child record of <i>pRecord</i> specified in <i>param1</i> . |
| CMA_LAST       | Last record in the container.                                     |
| CMA_LASTCHILD  | Last child record of <i>pRecord</i> specified in <i>param1</i> .  |
| CMA_NEXT       | Next record of <i>pRecord</i> specified in <i>param1</i> .        |
| CMA_PARENT     | Parent of <i>pRecord</i> specified in <i>param1</i> .             |
| CMA_PREV       | Previous record of <i>pRecord</i> specified in <i>param1</i> .    |

##### fsSearch (USHORT)

Enumeration order.

Specifies the enumeration order. This value is one of the following:

|               |  |
|---------------|--|
| CMA_ITEMORDER | Container records are enumerated in item order, first to last.   |
| CMA_ZORDER    | Container records are enumerated by z-order, from first record in the z-order to the last record. The last z-order record is the last record to be drawn. This flag is valid for the icon view only. |



## Returns

### **pRecord** (RECORDCORE)

Pointer to the RECORDCORE structure for which data was requested.

NULL No RECORDCORE structures to retrieve.

-1 The container record data was not returned. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

Other Pointer to the container record for which data was requested.

## Remarks

If the *cmd* parameter has the value of CMA\_FIRST or CMA\_LAST, the *pRecord* parameter in *param1* is ignored and the first or last record, respectively, in the container is returned.

Depending on the value of the *fsSearch* parameter, the container records are enumerated in item order or in z-order.

See "RECORDCORE" on page A-175 or "MINIRECORDCORE" on page A-124 for a complete list and descriptions of all container record attributes.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_QUERYRECORDEMPHASIS

This message queries for a container record with the specified emphasis attributes.

## Parameters

### **param1**

#### **pSearchAfter** (RECORDCORE)

Pointer to the specified container record.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

The values of this parameter can be:

CM\_FIRST Start the search with the first record in the container.

Other Start the search after the record specified by this pointer.

## param2

### **fEmphasisMask** (USHORT)

Emphasis attribute.

Specifies the emphasis attribute of the container record. The following states can be combined using a logical OR operator (|):

|                |  |
|----------------|--|
| CRA_COLLAPSED  | Specifies that a record is collapsed.  |
| CRA_CURSORED   | Specifies that a record will be drawn with a selection cursor.                 |
| CRA_DISABLED   | Specifies that a record will be drawn with unavailable-state emphasis.         |
| CRA_DROPONABLE | Specifies that a record can be a target for direct manipulation.               |
| CRA_EXPANDED   | Specifies that a record is expanded.   |
| CRA_FILTERED   | Specifies that a record is filtered and, therefore, hidden from view.          |
| CRA_INUSE      | Specifies that a record will be drawn with in-use emphasis.                    |
| CRA_PICKED     | Specifies that the container record will be picked up as part of the drag set. |
| CRA_SELECTED   | Specifies that a record will be drawn with selected-state emphasis.            |
| CRA_SOURCE     | Specifies that a record will be drawn with source-menu emphasis.               |

## Returns

### **pRecord** (PRECORDCORE)

Pointer to the record with the specified emphasis.

NULL This implies that none of the records that follow the pointer specified in the *pSearchAfter* parameter meet those specifications.

-1 The container record data was not returned.

The WinGetLastError function may return the following error:

### **PMERR\_INVALID\_PARAMETERS (1208)**

Other Pointer to a container record with the specified emphasis.

This is the first record that follows the record pointed to by the *pSearchAfter* parameter and satisfies the criteria specified in the *fEmphasisMask* parameter. To find the next record that satisfies this criteria, send this message again, but this time use the value returned in the *pRecord* parameter for the value of the *pSearchAfter* parameter.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_QUERYRECORDFROMRECT

This message queries for a container record that is bounded by the specified rectangle.

### Parameters

#### param1

##### **pSearchAfter** (PRECORDCORE)

Pointer to the specified container record.

To get all the container records within the specified rectangle, this message is sent repeatedly, each time this parameter is set to the pointer that is returned by the previous usage of this message.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

The values of this parameter can be:

|           |  |
|-----------|--|
| CMA_FIRST | Start the search with the first record in the container.     |
| Other     | Start the search after the record specified by this pointer. |

#### param2

##### **pQueryRecFromRect** (PQUERYRECFROMRECT)

Pointer to the QUERYRECFROMRECT data structure.

See "QUERYRECFROMRECT" on page A-173 for definitions of this structure's fields as they apply to the CM\_QUERYRECORDFROMRECT message.

### Returns

#### **pRecord** (PRECORDCORE)

Pointer to the container records within the bounding rectangle.

NULL No container records are within the bounding rectangle.

-1 The container record data was not returned. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

Other Pointer to the container record within the bounding rectangle.

## Remarks

This message returns the pointer to the first container record found in the rectangle after the starting position specified in the *pSearchAfter* parameter.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_QUERYRECORDINFO

This message updates the specified records with the current information for the container.

## Parameters

### param1

#### **pRecordArray** (PVOID)

Pointer to an array of pointers to RECORDCORE structures to which the current information is to be copied.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE all applicable data structures and messages.

### param2

#### **cNumRecord** (USHORT)

Number of records.

The number of container records to be updated. If the *cNumRecord* parameter has a value of 0, all of the records in the container are updated and the *pRecordArray* parameter is ignored.

## Returns

### **rc** (BOOL)

Success indicator.

TRUE Record information was successfully updated.

FALSE An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

## Remarks

This message is needed only if the application is sharing records among multiple containers in the same process.

The *filRecordAttr* and *ptllcon* fields are updated internally when they change, but not in the external RECORDCORE structure. Therefore, the application's external record does not always have current information in these fields. This message is only needed if the application is sharing records among multiple containers in the same process.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_QUERYRECORDRECT

This message returns the rectangle of the specified container record, relative to the container window origin.

### Parameters

**param1**

**prcllItem (PRECTL)**

Pointer to the RECTL structure, into which the rectangular coordinates are placed.

**param2**

**pQueryRecordRect (PQUERYRECORDRECT)**

Pointer to the QUERYRECORDRECT structure.

See "QUERYRECORDRECT" on page A-174 for definitions of this structure's fields as they apply to the CM\_QUERYRECORDRECT message.

### Returns

**rc (BOOL)**

Success indicator.

**TRUE** A rectangle with valid coordinates is returned.

**FALSE** The rectangle is not successfully returned. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Remarks

The coordinates of the returned rectangle are in window coordinates.

If the input record is not found in the container, the output rectangle is empty.

For a container using the details view (CV\_DETAIL), all of the data for a row is returned in the rectangle.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_QUERYVIEWPORTRECT

This message returns a rectangle that contains the coordinates of the container's client area. These are virtual coordinates that are relative to the origin of the coordinate space requested.

### Parameters

**param1**

**prclViewport** (PRECTL)

Pointer to the RECTL structure.

Pointer to the RECTL structure that the virtual coordinates of the client area rectangle are to be written into.

**param2**

**usIndicator** (USHORT)

Coordinate space indicator.

One of the following must be used:

**CMA\_WINDOW** Returns the client area rectangle in container window coordinates.

**CMA\_WORKSPACE** Return the client area rectangle in coordinates relative to the origin of the container's workspace.

**fRightSplitWindow** (BOOL)

Flag.

Flag that specifies the right or left window in the split details view. This flag is ignored if the view is not the split details view.

**TRUE** Right split window is returned.

**FALSE** Left split window is returned.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** The client area rectangle was returned successfully.

**FALSE** An error occurred. The `WinGetLastError` function may return the following error:

`PMERR_INVALID_PARAMETERS`.

## Remarks

The virtual coordinates of the client area rectangle are written into the structure addressed by the *prclViewport* parameter.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `FALSE`.

---

## CM\_REMOVEDTAILFIELDINFO

This message removes one, multiple, or all `FIELDINFO` structures from the container control.

## Parameters

**param1**

**pFieldInfoArray** (PVOID)

Pointer to an array of pointers to `FIELDINFO` structures that are to be removed.

**param2**

**cNumFieldInfo** (USHORT)

Number of `FIELDINFO` structures to be removed.

If the *cNumFieldInfo* parameter has a value of 0, all of the `FIELDINFO` structures in the container are removed and the *pFieldInfoArray* parameter is ignored.

**fRemoveFieldInfo** (USHORT)

Flags.

Flags that show whether memory must be freed and `FIELDINFO` structures invalidated.

`CMA_FREE`

If specified, `FIELDINFO` structures are removed and memory associated with the `FIELDINFO` structures is freed. If not specified, `FIELDINFO` structures are removed and no memory is freed; this is the default.

**CMA\_INVALIDATE** If specified, after FIELDINFO structures are removed, the container is invalidated, and any necessary repositioning of the FIELDINFO structures is performed. If not specified, invalidation is not performed.

## Returns

### **cFields** (SHORT)

Number of structures.

-1 An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_MEMORY\_DEALLOCATION\_ERR.

Other The number of FIELDINFO structures that remain in the container.

## Remarks

The FIELDINFO structures are removed from the list of columns inserted into the container control.

If the CMA\_FREE attribute is not specified, the container control removes the specified FIELDINFO structures without freeing the memory. The application is responsible for freeing the memory associated with the FIELDINFO structures by using the CM\_FREEDTAILFIELDINFO message.

If the *cNumFieldInfo* parameter has a value of 0 and the CMA\_FREE attribute is specified, all of the FIELDINFO structures in the container control are removed and the memory associated with the FIELDINFO structures is freed. It is the application's responsibility to free all of the application-allocated memory associated with the FIELDINFO structures.

If the number of pointers to FIELDINFO structures in the array exceeds the count of FIELDINFO structures to be removed, only the number of structures specified in the *cNumFieldInfo* parameter are removed. If the CCS\_VERIFYPOINTERS style bit is set and the *pFieldInfoArray* parameter contains pointers to a FIELDINFO structure or structures that do not exist, the PMERR\_INVALID\_PARAMETERS error is set.

If you do not want to show a column, you can hide it by setting the CFA\_INVISIBLE attribute of the FIELDINFO data structure and notifying the container control with the CM\_INVALIDATEDTAILFIELDINFO message.

If the CMA\_INVALIDATE attribute is specified, the container is repainted.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.



---

## CM\_REMOVECORE

This message removes one, multiple, or all RECORDCORE structures from the container control.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Parameters

**param1**

**pRecordArray** (PVOID)

Pointer to an array of pointers to RECORDCORE structures that are to be removed.

**param2**

**cNumRecord** (USHORT)

Number of records.

Number of container records to be removed. If the *cNumRecord* parameter has a value of 0, all of the records in the container are removed and the *pRecordArray* parameter is ignored.

**fRemoveRecord** (USHORT)

Flags.

Flags that show whether memory must be freed and container records invalidated.

CMA\_FREE

If specified, RECORDCORE structures are removed and memory associated with the RECORDCORE structures is freed. If not specified, RECORDCORE structures are removed and no memory is freed; this is the default.

CMA\_INVALIDATE

If specified, after RECORDCORE structures are removed the container is invalidated and any necessary repositioning of the container records is performed. If not specified, invalidation is not performed.

This option is not valid in the icon view unless the CCS\_AUTOPOSITION style bit is not set. In the icon view, the container record is refreshed if the CCS\_AUTOPOSITION style bit is set, regardless of whether the CMA\_INVALIDATE attribute is set.

## Returns

### **cRecords** (LONG)

Number of structures.

-1 An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_MEMORY\_DEALLOCATION\_ERR.

Other Number of root level RECORDCORE structures that remain in the container.

## Remarks

When parent item records are removed, all associated child item records are removed, as well.

If the CMA\_FREE attribute is not specified, the container control removes the specified RECORDCORE structures without freeing the memory. The application is responsible for freeing the memory associated with the RECORDCORE structure by using the CM\_FREERECORD message.

If the *cNumRecord* parameter has a value of 0 and the CMA\_FREE attribute is specified, all of the RECORDCORE structures in the container control are removed and the memory associated with the RECORDCORE structures is freed. It is the application's responsibility to free all of the application-allocated memory associated with the RECORDCORE structures.

If the number of pointers to RECORDCORE structures in the array exceeds the count of RECORDCORE structures to be removed, only the number of records specified in the *cNumRecord* parameter is removed. If the CCS\_VERIFYPOINTERS style bit is set and the *pRecordArray* parameter contains pointers to a RECORDCORE structure or structures that do not exist, the PMERR\_INVALID\_PARAMETERS error is set.

If the CMA\_INVALIDATE attribute is specified, the container is repainted if the removed record or records are visible.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## CM\_SCROLLWINDOW

This message scrolls an entire container window.

### Parameters

**param1**

**fsScrollDirection** (USHORT)

Scroll direction.

Direction in which to scroll the container window.

CMA\_VERTICAL      Scroll vertically.

CMA\_HORIZONTAL    Scroll horizontally.

**param2**

**IScrollInc** (LONG)

Scroll increment.

Amount (in pixels) by which to scroll the window.

### Returns

**rc** (BOOL)

Success indicator.

TRUE      Successful completion

FALSE     An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Remarks

If the *IScrollInc* parameter value is greater than 0 and the CMA\_HORIZONTAL attribute is specified, the container window is scrolled to the right. The container window is scrolled down if the *IScrollInc* parameter value is greater than 0 and the CMA\_VERTICAL attribute is specified. Similarly, the container window is scrolled left and up, respectively, if the *IScrollInc* parameter value is less than 0 and the same two attributes are specified.

If you want the container window to be scrolled by an amount that is indicated with a key, such as the PgUp, PgDn, Home, and End keys, the application can send a key event to the scroll bar.

If the container window is displaying the split details view, the CM\_HORZSCROLLSPLITWINDOW message is used for horizontal scrolling.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_SEARCHSTRING

This message returns the pointer to a container record whose text matches the string.

### Parameters

**param1**

**pSearchString** (PSEARCHSTRING)

Pointer to the SEARCHSTRING structure.

See "SEARCHSTRING" on page A-184 for definitions of this structure's fields as they apply to the CM\_SEARCHSTRING message.

**param2**

**pSearchAfter** (PRECORDCORE)

Pointer to the starting container record.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

**CMA\_FIRST** Start the search at the first container record.

**Other** Start the search after the container record specified by this pointer. To get all of the records in the container whose text matches the string, this message is sent repeatedly. Each time this message is sent, the *pSearchAfter* parameter contains a pointer to the last record that was found.

### Returns

**pRecord** (PRECORDCORE)

Pointer to the found container record.

**NULL** No container record's text matches the search string.

**-1** An error occurred. The WinGetLastError function may return the following error:

**PMERR\_INVALID\_PARAMETERS.**

**Other** Pointer to the container record whose text matches the search string.

### Remarks

The CM\_SEARCHSTRING message is NLS-enabled.

In the details view, the string is searched for in each column of each record.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULL.

---

## CM\_SETCNRINFO

This message sets or changes the data for the container control.

### Parameters

#### param1

##### pCnrInfo (PCNRINFO)

Pointer to the CNRINFO structure from which to set the data for the container.

#### param2

##### uiCnrInfoFl (ULONG)

Flags.

Flags that show which fields are to be set.

##### CMA\_PSORTRECORD

Pointer to the comparison function for sorting container records. If NULL, which is the default condition, no sorting is performed. Sorting only occurs during record insertion and when changing the value of this field. The third parameter of the comparison function, *pStorage*, must be NULL. See CM\_SORTRECORD for a further description of the comparison function.

##### CMA\_PFIELDINFOLAST

Pointer to the last column in the left window of the split details view. The default is NULL, causing all columns to be positioned in the left window.

##### CMA\_PFIELDINFOBJECT

Pointer to a column that represents an object in the details view. This FIELDINFO structure must contain icons or bit maps. In-use emphasis is applied to this column of icons or bit maps only. The default is the leftmost column in the unsplit details view, or the leftmost column in the left window of the split details view.

##### CMA\_CNRTITLE

Text for the container title. The default is NULL.

##### CMA\_FLWINDOWATTR

Container window attributes.

##### CMA\_PTLORIGIN

Lower-left origin of the container window in virtual workspace coordinates, used in the icon view. The default origin is **(0,0)**.

##### CMA\_DELTA

An application-defined threshold, or number of records, from either end of the list of available records. Used when a container needs to handle large amounts of data. The default is 0. Refer to the description of the container control

|                        |  |
|------------------------|--|
|                        | in the <i>OS/2 Programming Guide</i> for more information about specifying deltas.   |
| CMA_SLBITMAPORICON     | The size (in pels) of icons or bit maps. The default is the system size.   |
| CMA_SLTREEBITMAPORICON | The size (in pels) of the expanded and collapsed icons or bit maps in the tree icon and tree text views.   |
| CMA_TREEBITMAP         | Expanded and collapsed bit maps in the tree icon and tree text views.  |
| CMA_TREEICON           | Expanded and collapsed icons in the tree icon and tree text views.   |
| CMA_LINESPACING        | The amount of vertical space (in pels) between the records. If this value is less than 0, a default value is used.   |
| CMA_CXTREEINDENT       | Horizontal distance (in pels) between levels in the tree view. If this value is less than 0, a default value is used.  |
| CMA_CXTREELINE         | Width of the lines (in pels) that show the relationship between items in the tree view. If this value is less than 0, a default value is used. Also, if the CA_TREELINE container attribute of the CNRINFO data structure's <i>flWindowAttr</i> field is not specified, these lines are not drawn. |
| CMA_XVERTSPLITBAR      | The initial position of the split bar relative to the container, used in the details view. If this value is less than 0, the split bar is not used. The default value is negative one (-1).  |

**rc** (BOOL)

Success indicator.

TRUE Container data was successfully set.

FALSE Container data was not set. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_INSUFFICIENT\_MEMORY.

**Remarks**

The data for a container is set from the buffer addressed by the *pCnrInfo* parameter. The flags in the *ulCnrInfoFl* parameter show which part or parts of the *pCnrInfo* parameter are set. The flag values can be combined by using a logical OR operator (|).

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## CM\_SETRECORDEMPHASIS

This message sets the emphasis attributes of the specified container record.

### Parameters

**param1**

#### **pRecord** (PCORECORE)

Pointer to the specified container record.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PCORECORE in all applicable data structures and messages.

**param2**

#### **usChangeEmphasis** (USHORT)

Change-emphasis-attribute flag.

**TRUE** The container record's emphasis attribute is to be set ON if the change specified is not the same as the current state.

**FALSE** The container record's emphasis attribute is to be set OFF if the change specified is not the same as the current state.

#### **fEmphasisAttribute** (USHORT)

Emphasis attribute of the container record.

The following states can be combined by using a logical OR operator (|):

**CRA\_CURSORED** Specifies that a record will be drawn with a selection cursor.

**CRA\_DISABLED** Specifies that a record will be drawn with unavailable-state emphasis.

**CRA\_INUSE** Specifies that a record will be drawn with in-use emphasis.

**CRA\_PICKED** Specifies that the container record will be picked up as part of the drag set.

**CRA\_SELECTED** Specifies that a record will be drawn with selected-state emphasis.

**CRA\_SOURCE** Specifies that a record will be drawn with source-menu emphasis.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE An error occurred.

The WinGetLastError function may return the following errors:

**PMERR\_INVALID\_PARAMETERS (1208)**

**PMERR\_INSUFFICIENT\_MEMORY (203E)**

## Remarks

For single-selection containers, the selection of the previous container record is cancelled before another record is selected. The selection cursor is set with the CRA\_CURSORED attribute for single-selection containers. Only one selection cursor is allowed.

The selection cursor must always be available to the user. Therefore, if you attempt to disable the selection cursor by specifying FALSE for the *usChangeEmphasis* parameter and CRA\_CURSORED for the *fEmphasisAttribute* parameter, the PMERR\_INVALID\_PARAMETERS error is set. In order to change the selection cursor attribute, TRUE should be specified for the *usChangeEmphasis* parameter and CRA\_CURSORED for the *fEmphasisAttribute* parameter. The *pRecord* parameter should point to the record to which the selection cursor should be applied. The container control removes the selection cursor from the record with the cursor and applies it to the new record.

A CN\_EMPHASIS notification code is sent to the container owner if the record emphasis attribute is changed.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.



---

## CM\_SORTRECORD

This message sorts the container records in the container control.

### Parameters

#### param1

##### **pfnCompare** (PFN)

Pointer to a comparison function.

#### param2

##### **pStorage** (PVOID)

Application use.

Available for application use.

### Returns

#### rc (BOOL)

Success indicator.

TRUE The records in the container were sorted.

FALSE The records in the container were not sorted. The WinGetLastError function may return the following errors:

- PMERR\_COMPARISON\_FAILED
- PMERR\_INSUFFICIENT\_MEMORY.

### Remarks

The *pfnCompare* parameter must be declared as:

```
SHORT EXPENTRY pfnCompare(RECORDCORE p1, RECORDCORE p2, PVOID pStorage);
```

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages.

The *pfnCompare* parameter points to an application-provided function that compares two RECORDCORE structures and returns a SHORT value that specifies their relationship. The *pfnCompare* parameter is called one or more times during the sorting process and is passed pointers to two RECORDCORE structures on each call. The routine must compare the RECORDCORE structures, and then return one of the following values:

| <u>Value</u> | <u>Meaning</u>                        |
|--------------|---------------------------------------|
| >0           | <i>p1</i> is less than <i>p2</i> .    |
| 0            | <i>p1</i> is equal to <i>p2</i> .     |
| <0           | <i>p1</i> is greater than <i>p2</i> . |

The container records are sorted in increasing order, as defined by the *pfnCompare* parameter. The records can be sorted in reverse order by reversing the sense of “greater than” and “less than” in the *pfnCompare* parameter.

If the container has only one record, the `PMERR_COMPARISON_FAILED` error is set.

The application must provide an NLS-enabled function for the *pfnCompare* parameter. The container control does not provide NLS enablement for sorting.

An alternative to using the `CM_SORTRECORD` message is to provide an application-defined comparison function to sort the container records, which can be specified in the `CNRINFO` structure's *pSortRecord* field. If this function is provided, the container records are sorted as they are inserted into the container control. If this field is `NULL`, the records are not sorted on insertion.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `FALSE`.

---

## CM\_SETTEXTVISIBILITY

This message sets the visibility state of text for records in the container control.

### Parameters

#### param1

##### **bVisible** (BOOL)

Text visibility state.

TRUE    Text is visible.

FALSE   Text is not visible.

#### param2

##### **Reserved** (PVOID)

Reserved value, 0.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE    Text visibility state was successfully set.

FALSE   Error occurred.

## Remarks

This message is used to set the visibility state of the text for records in the container control. If *bVisible* is TRUE, text will appear with the icons in icon view, name view, tree icon view, and tree name view. If *bVisible* is FALSE, no text appears.

This message does not apply to any variation of text view (icon text, tree text) or details view.

This message affects ALL records within the container. The visibility state of the text cannot be set for individual records.

## Default Processing

The default window procedure does not expect to receive this message and, therefore, takes no action on it other than to return FALSE.

---

## WM\_PICKUP

This message adds objects to the drag set during a lazy drag operation.

## Parameters

**param1**

**ptlPointerPos** (POINTL)

Pointer position in window coordinates relative to the bottom-left corner of the window.

**param2**

**Reserved** (ULONG)

Reserved value, must be 0.

## Returns

**returns**

**rc** (BOOL)

Success indicator.

Possible values are described in the following list:

TRUE    Message was processed.

FALSE   Message was ignored.

## Remarks

This message will be posted to the application queue associated with the window that has the focus, or with the window that is to receive the pointer-button information.

WM\_PICKUP message is sent to the window under the mouse pointer when the user presses the direct-manipulation button while holding down the lazy drag augmentation key, currently the ALT key. This message is used to inform an application that the user is commencing a lazy drag operation. The container control sends its owner a CN\_PICKUP notification when it receives a message.

Objects are added to the drag set when a WM\_PICKUP message is received. The first time the message is received, the application initiates a lazy drag operation. Each subsequent WM\_PICKUP message that is received during the course of the lazy drag operation indicates that objects are to be added to the drag set.

### Default Processing

The default message procedure sets *rc* to TRUE.

---

## WM\_PRESPARAMCHANGED (in Container Controls)

For the cause of this message, see WM\_PRESPARAMCHANGED.

### Parameters

**param1**

**attrtype** (ULONG)

Presentation parameter attribute identity.

PP\_BACKGROUNDCOLOR or PP\_BACKGROUNDINDEX

Sets the background color of the container window. This color is initially set to SYSCLR\_WINDOW.

PP\_BORDERCOLOR or PP\_BORDERINDEX

Sets the color of the title separators, column separators, and split bar. This color is initially set to SYSCLR\_WINDOWFRAME.

PP\_FONTNAMESIZE

Sets the font and font size of the text in the container. This font and font size defaults to the system font and font size.

PP\_FOREGROUNDINDEX or PP\_FOREGROUNDINDEX

Sets the color of unselected text. This color is initially set to SYSCLR\_WINDOWTEXT.

PP\_HILITEBACKGROUNDINDEX or PP\_HILITEBACKGROUNDINDEX

Sets the color of selection emphasis, the color of the cursor of an unselected item in the details view, and the color of the cursor in all other views. This color is initially set to SYSCLR\_HILITEBACKGROUND.

PP\_HILITEFOREGROUNDINDEX or PP\_HILITEFOREGROUNDINDEX

Sets the color of the text of a selected item in all views and the color of the cursor of a selected item in the details view. This color is initially set to SYSCLR\_HILITEFOREGROUND.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

**ulReserved (ULONG)**

Reserved value, should be 0.

### **Remarks**

The application uses the WinSetPresParam function to change presentation parameters. This results in a WM\_PRESPARAMCHANGED (in Container Controls) message being sent to the container.

### **Default Processing**

For a description of the default processing, see WM\_PRESPARAMCHANGED.

---

## Chapter 23. Notebook Control Window Processing

This system-provided window procedure processes the actions on a notebook control (WC\_NOTEBOOK).

---

### Purpose

A notebook control (WC\_NOTEBOOK window class) is a visual component whose specific purpose is to organize information on individual pages so that a user can find and display that information quickly and easily. It simulates a real-world notebook while improving it by overcoming its natural limitations. A user can select and display pages by using either a pointing device, such as a mouse, or the keyboard.

The notebook is designed to be customizable to meet varying application requirements, while providing an easy-to-use user interface component that can be used to develop products that conform to the Common User Access\* (CUA\*) user interface guidelines. The application can specify different colors, sizes, and orientations for its notebooks, but the underlying function of the control remains the same. For a complete description of CUA notebooks, refer to the *SAA CUA Guide to User Interface Design* and the *SAA CUA Advanced Interface Design Reference*.

---

### Notebook Control Styles

Notebook control window styles can be set when a notebook is created. The following styles can be set when creating a notebook control window. If no styles are specified, defaults, which are identified in the following descriptions, are used.

- Specify one of the following to determine whether the control is a solid bound or spiral bound notebook, or a catalog:

**BKS\_SOLIDBIND**                      Paints a solid binding on the notebook. This is the default.

**BKS\_SPIRALBIND**                      Paints a spiral binding on the notebook.

- Specify one of the following to determine where the back pages are positioned:

**BKS\_BACKPAGESBR**                      Paints back pages on the notebook's bottom and right sides. This is the default.

**BKS\_BACKPAGESBL**                      Paints back pages on the notebook's bottom and left sides.

**BKS\_BACKPAGESTR**                      Paints back pages on the notebook's top and right sides.

**BKS\_BACKPAGESTL**                      Paints back pages on the notebook's top and left sides.

- Specify one of the following to determine the side of the notebook on which the major tabs are positioned. Valid combinations with back pages styles are noted in each definition.

**BKS\_MAJORTABRIGHT** Places major tabs on the notebook's right edge. Only valid in combination with BKS\_BACKPAGESBR or BKS\_BACKPAGESTR. This is the default when either of these back pages styles is used.

**BKS\_MAJORTABLEFT** Places major tabs on the notebook's left edge. Only valid in combination with BKS\_BACKPAGESBL or BKS\_BACKPAGESTL. This is the default when BKS\_BACKPAGESTL is used.

**BKS\_MAJORTABTOP** Places major tabs on the notebook's top edge. Only valid in combination with BKS\_BACKPAGESTR or BKS\_BACKPAGESTL.

**BKS\_MAJORTABBOTTOM** Places major tabs on the notebook's bottom edge. Only valid in combination with BKS\_BACKPAGESBR or BKS\_BACKPAGESBL. This is the default when BKS\_BACKPAGESBL is used.

- Specify one of the following to set the shape of the notebook tabs:

**BKS\_SQUARETABS** Draws tabs with square edges. This is the default.

**BKS\_ROUNDEDTABS** Draws tabs with rounded edges.

**BKS\_POLYGONTABS** Draws tabs with polygon edges.

- Specify one of the following to position the status line text:

**BKS\_STATUSTEXTLEFT** Left-justifies status line text. This is the default.

**BKS\_STATUSTEXTRIGHT** Right-justifies status line text.

**BKS\_STATUSTEXTCENTER** Centers status line text.

- Specify one of the following to position the tab text:

**BKS\_TABTEXTCENTER** Centers tab text. This is the default.

**BKS\_TABTEXTLEFT** Left-justifies tab text.

**BKS\_TABTEXTRIGHT** Right-justifies tab text.

---

## Notebook Control Data

See the following for descriptions of the notebook control data structures:

- "BOOKTEXT" on page A-23
- "DELETENOTIFY" on page A-46
- "PAGESELECTNOTIFY" on page A-140.

---

## Notebook Control Notification Messages

These messages are initiated by the notebook control window to notify its owner of significant events.

---

### WM\_CONTROL (in Notebook Controls)

For the cause of this message, see WM\_CONTROL.

#### Parameters

##### param1

###### id (USHORT)

Control-window identity.

###### notifycode (USHORT)

Notify code.

The notebook control uses these notification codes:

|                         |   |
|-------------------------|---|
| BKN_HELP                | Indicates the notebook control has received a WM_HELP message.  |
| BKN_NEWPAGESIZE         | Indicates the dimensions of the application page window have changed.   |
| BKN_PAGEDELETED         | Indicates a page has been deleted from the notebook.  |
| BKN_PAGESELECTED        | Indicates a new page has been brought to the top of the notebook. This notification is sent after the page is turned.   |
| BKN_PAGESELECTEDPENDING | Indicates a new page is about to be brought to the top of the notebook. This notification is sent before the page is actually turned.<br><br>If the application does not want the page to be turned, it sets the <i>ulPageIdNew</i> field of the PAGESELECTNOTIFY structure to NULL before returning. |

##### param2

###### notifyinfo (ULONG)

Notify code information.

The value of this parameter depends on the value of the *notifycode* parameter. When the value of the *notifycode* parameter is BKN\_HELP, this parameter is the ID of the notebook page (*ulPageId*) whose tab contains the selection cursor.



When the value of the *notifycode* parameter is BKN\_PAGESELECTED or BKN\_PAGESELECTEDPENDING, this parameter is a pointer to the PAGESELECTNOTIFY structure.

When the value of the *notifycode* parameter is BKN\_PAGEDELETED, this parameter is a pointer to the DELETENOTIFY structure.

Otherwise, this parameter is the notebook control window handle.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The notebook control window procedure generates this message and sends it to its owner, informing the owner of this event.

## Default Processing

For a description of the default processing, see WM\_CONTROL.

---

## WM\_CONTROLPOINTER (in Notebook Controls)

For the cause of this message, see WM\_CONTROLPOINTER.

For a description of the parameters, see WM\_CONTROLPOINTER.

## Remarks

For the appropriate remarks, see WM\_CONTROLPOINTER.

## Default Processing

For the default processing, see WM\_CONTROLPOINTER.

---

## WM\_DRAWITEM (in Notebook Controls)

This notification message is sent to the owner of a notebook control each time a tab's content is to be drawn by the owner of the notebook. The tab's content is drawn by the owner unless the owner sets the tab text or bit map by sending a BKM\_SETTABTEXT or BKM\_SETTABBITMAP message, respectively, to the notebook control.

## Parameters

**param1**

**id** (USHORT)

Window identifier.

The window identifier of the notebook control sending this notification message.

## param2

### **powneritem** (POWNERITEM)

Pointer to an OWNERITEM data structure.

The following list defines the OWNERITEM data structure fields that apply to the notebook control. See "OWNERITEM" on page A-136 for the default field values.

#### *hwnd* (HWND)

Notebook window handle.

#### *hps* (HPS)

Presentation-space handle.

#### *fsState* (ULONG)

Notebook window style flags. See "Notebook Control Styles" on page 23-1 for descriptions of these style flags.

#### *fsAttribute* (ULONG)

Page attribute flags for the tab page. See BKM\_INSERTPAGE for descriptions of these attribute flags.

#### *fsStateOld* (ULONG)

Reserved.

#### *fsAttributeOld* (ULONG)

Reserved.

#### *rcItem* (RECTL)

Tab rectangle to be drawn in window coordinates.

#### *idlItem* (LONG)

Reserved.

#### *hlItem* (ULONG)

Current page ID (*ulPageId*) for which the content of a tab is to be drawn.

## **Returns**

### **rc** (BOOL)

Content-drawn indicator.

**TRUE** The owner draws the tab's content.

**FALSE** If the owner does not draw the tab's content, the owner returns this value and the notebook control draws the tab's content.

## **Remarks**

If an application uses notebook controls that contain tab pages, the default condition is for the application to draw the contents of the tab each time a tab page is displayed. This situation applies particularly if the content of the tab is not one of the supported formats.

The notebook control window procedure generates this message and sends it to its owner, informing the owner that the content of a tab is to be drawn. The owner is given the opportunity to draw the content of the tab and to indicate that the content of the tab has been

drawn or that the notebook control is to draw it. To indicate that the notebook control is to draw the content of the tab, the owner sends either a `BKM_SETTABTEXT` or a `BKM_SETTABBITMAP` message to the notebook control.

### **Default Processing**

For a description of the default processing, see `WM_DRAWITEM`.

---

## Notebook Control Window Messages

This section describes the notebook control window procedure actions on receiving the following messages.

---

### BKM\_CALCPAGERECT

This message calculates an application page rectangle from a notebook rectangle or calculates a notebook rectangle from an application page rectangle, depending on the setting of the *bPage* parameter.

#### Parameters

**param1**

**pRectl** (RECTL)

Pointer to the RECTL structure that contains the coordinates of the rectangle.

If the *bPage* parameter is TRUE, this structure contains the coordinates of a notebook window on input, and on return it contains the coordinates of an application page window.

If the *bPage* parameter is FALSE, this structure contains the coordinates of an application page window on input, and on return it contains the coordinates of a notebook window.

**param2**

**bPage** (BOOL)

Window specifier.

Specifies whether the window coordinates to calculate are for a notebook window or an application page window.

TRUE    An application page window is calculated.

FALSE   A notebook window is calculated.

#### Returns

**rc** (BOOL)

Success indicator.

TRUE    Coordinates were successfully calculated.

FALSE   Unable to calculate coordinates. This is returned if an invalid RECTL structure is specified in the *pRectl* parameter.

#### Remarks

The application can use this message to determine the size of either the notebook window or the application page window. It can also be used when the application handles the position and size of the application page window.

To calculate the application page rectangle, specify the coordinates of the notebook window in the *pRectl* parameter and TRUE in the *bPage* parameter. The notebook control then uses the coordinates specified in the *pRectl* parameter to calculate and return the coordinates of the application page window.

To calculate the notebook rectangle, specify the coordinates of the application page window in the *pRectl* parameter and FALSE in the *bPage* parameter. The notebook control then uses the coordinates specified in the *pRectl* parameter to calculate and return the coordinates of the notebook window.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_DELETEPAGE

This message deletes the specified page or pages from the notebook data list.

### Parameters

**param1**

**ulPageId** (ULONG)

Page identifier.

Page identifier for deletion. This is ignored if the BKA\_ALL attribute of the *usDeleteFlag* parameter is specified.

**param2**

**usDeleteFlag** (USHORT)

Page range attribute.

Attribute that specifies the range of pages to be deleted.

**BKA\_SINGLE** Delete a single page.

**BKA\_TAB** If the page ID specified is that of a page with a major tab attribute, delete that page and all subsequent pages up to the next page that has a major tab attribute.

If the page ID specified is that of a page with a minor tab attribute, delete that page and all subsequent pages up to the next page that has either a major or minor tab attribute.

This attribute should only be specified for pages that have major or minor tab attributes. If a page with neither of these attributes is specified, FALSE is returned and no pages are deleted.

**BKA\_ALL** Delete all pages in the notebook.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Pages were successfully deleted.

**FALSE** Unable to delete the page or pages. This is returned if an invalid page ID is specified for the *ulPageId* parameter or if the **BKA\_TAB** attribute is specified for a page that has neither a major nor a minor tab attribute.

## Remarks

The notebook frees all storage that it has allocated for the deleted page or pages. The application is responsible for deleting the application page window and bit map, if created.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return **FALSE**.

---

## BKM\_INSERTPAGE

This message inserts the specified page into the notebook data list.

## Parameters

**param1**

**ulPageId** (ULONG)

Page ID for placement.

Page identifier used for the placement of the inserted page. This identifier is ignored if the **BKA\_FIRST** or **BKA\_LAST** attribute of the *usPageOrder* parameter is specified.

**param2**

**usPageStyle** (USHORT)

Style attributes.

Attributes that specify the style to be used for an inserted page. You can specify one attribute from each of the following groups by using logical OR operators (**|**) to combine attributes.

- Specify the following for automatic page position and size:

**BKA\_AUTOPAGESIZE** Notebook handles the positioning and sizing of the application page window specified in the **BKM\_SETPAGEWINDOWHWND** message.

- Specify the following to display status area text:

**BKA\_STATUSTEXTON**

Page is to be displayed with status area text. If this attribute is not specified, the application cannot associate a text string with the status area of the page being inserted.

- Specify one of the following if the page is to have a major or minor tab attribute:

|           |  |
|-----------|--|
| BKA_MAJOR | Inserted page will have a major tab attribute. |
| BKA_MINOR | Inserted page will have a minor tab attribute. |

**usPageOrder (USHORT)**

Order attributes.

Placement of page relative to the previously inserted pages. You can specify one of the following attributes:

|           |  |
|-----------|--|
| BKA_FIRST | Insert page at the front of the notebook. The page ID specified in the <i>ulPageld</i> parameter for <i>param1</i> is ignored if this is specified.  |
| BKA_LAST  | Insert page at the end of the notebook. The page ID specified in the <i>ulPageld</i> parameter for <i>param1</i> is ignored if this is specified.  |
| BKA_NEXT  | Insert page after the page whose ID is specified in the <i>ulPageld</i> parameter for <i>param1</i> . If the page ID specified in the <i>ulPageld</i> parameter is invalid, NULL is returned and no page is inserted.  |
| BKA_PREV  | Insert page before the page whose ID is specified in the <i>ulPageld</i> parameter for <i>param1</i> . If the page ID specified in the <i>ulPageld</i> parameter is invalid, NULL is returned and no page is inserted. |

**Returns**

**ulPageld (ULONG)**

Page ID for insertion.

Identifier for the inserted page.

NULL The page was not inserted into the notebook. An invalid page ID was specified for the *ulPageld* parameter for *param1* or not enough space was available to allocate the page data.

Other Identifier for the inserted page.

**Remarks**

The notebook control allocates and manages the storage needed for the new page. If neither the BKA\_MAJOR or BKA\_MINOR attribute is specified, the page is inserted with no tab attributes.

If the application does not specify the BKA\_AUTOPAGESIZE attribute, it must handle the positioning and sizing of the application page window when it receives the BKN\_NEWPAGESIZE notification code.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## BKM\_INVALIDATETABS

This message repaints all of the tabs in the notebook.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

TRUE    Tabs painted successfully.

FALSE    Tabs were not painted.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_QUERYPAGECOUNT

This message queries the number of pages.

### Parameters

**param1**

**ulPageId** (ULONG)

Page ID or 0.

Page identifier from which to start the query, or 0. If this parameter is set to 0, the query begins with the first page.



## param2

### usQueryEnd (USHORT)

Query end attribute.

Attribute that ends the page count query.

**BKA\_MAJOR** Query the number of pages between the page ID specified in the *ulPageld* parameter and the next page that has the **BKA\_MAJOR** attribute. The page that has the **BKA\_MAJOR** attribute is not included in the page count.

**BKA\_MINOR** Query the number of pages between the page ID specified in the *ulPageld* parameter and the next page that has the **BKA\_MINOR** attribute. The page that has the **BKA\_MINOR** attribute is not included in the page count.

**BKA\_END** Query the number of pages between the page ID specified in the *ulPageld* parameter and the last page. When this attribute is specified, the page count includes the last page plus the notebook's back cover.

## Returns

### pageCount (SHORT)

Number of pages.

Number of pages in the notebook.

**BOOKERR\_INVALID\_PARAMETERS** An invalid page ID was specified for the *ulPageld* parameter.

**Other** Number of pages for the specified range. If the notebook is empty or no pages are found in the range, this value is 0.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## BKM\_QUERYPAGEID

This message queries the 4 bytes of application reserved storage associated with the specified page.

### Parameters

#### param1

**ulPageId** (ULONG)

Page ID.

The page identifier of the page from which to retrieve the 4 bytes of data.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulPageData** (ULONG)

Page data.

Application-defined page data.

**BOOKERR\_INVALID\_PARAMETERS** An invalid page ID was specified for the *ulPageId* parameter.

**0** No page data was set for the page specified in the *ulPageId* parameter.

**Other** Application-defined page data.

### Remarks

This data is set by using the BKM\_SETPAGEDATA message.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## BKM\_QUERYPAGEID

This message queries the page identifier for the specified page.

## Parameters

### param1

#### **ulPageld** (ULONG)

Location page ID.

Page identifier used for locating the requested page. This identifier is ignored if the BKA\_FIRST, BKA\_LAST, or BKA\_TOP attribute is specified.

### param2

#### **usQueryOrder** (USHORT)

Page ID query order.

Order in which to query the page identifier.

**BKA\_FIRST** Get the page identifier for the first page. The page ID specified in the *ulPageld* parameter for *param1* is ignored if this is specified.

**BKA\_LAST** Get the page identifier for the last page. The page ID specified in the *ulPageld* parameter for *param1* is ignored if this is specified.

**BKA\_NEXT** Get the page identifier for the page after the page whose ID is specified in the *ulPageld* parameter for *param1*. If the page ID specified in the *ulPageld* parameter is invalid, BOOKERR\_INVALID\_PARAMETERS is returned.

**BKA\_PREV** Get the page identifier for the page before the page whose ID is specified in the *ulPageld* parameter for *param1*. If the page ID specified in the *ulPageld* parameter is invalid, BOOKERR\_INVALID\_PARAMETERS is returned.

**BKA\_TOP** Get the page identifier for the page currently visible in the notebook. The page ID specified in the *ulPageld* parameter for *param1* is ignored if this is specified.

#### **usPageStyle** (USHORT)

Page style.

Page style for which to query the page identifier. If neither of these attributes is specified, the *usPageStyle* parameter is ignored.

**BKA\_MAJOR** Query page with major tab attribute.

**BKA\_MINOR** Query page with minor tab attribute. If a major tab page is found before the minor tab page, the search is ended and 0 is returned.

## Returns

### **ulPageld** (ULONG)

Retrieved page ID.

|                            |  |
|----------------------------|--|
| BOOKERR_INVALID_PARAMETERS | Returned if the page ID specified for the <i>ulPageId</i> parameter for <i>param1</i> is invalid when specifying either the BKA_PREV or BKA_NEXT attribute in the <i>usQueryOrder</i> parameter. |
| 0                          | Requested page not found. This could be an indication that the end or front of the list has been reached, or that the notebook is empty.   |
| Other                      | Retrieved page identifier.   |

### Remarks

If the BKA\_FIRST, BKA\_LAST, or BKA\_TOP attribute is specified, the page ID in the *ulPageId* parameter is ignored.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

## BKM\_QUERYPAGEINFO

This message queries the page information associated with a notebook page.

### Parameters

**param1**

**ulPageId** (ULONG)

Id of the notebook page whose information is to be queried.

**param2**

**pPageInfo** (PPAGEINFO)

Pointer to a notebook page information structure.

## Returns

### returns

**rc** (BOOL)

Success indicator.

Possible values are described in the following list:

TRUE     Message was processed.  
FALSE    Message was ignored.

## Remarks

This message handles the following notebook messages:

- BKM\_QUERYPAGEID
- BKM\_QUERYPAGEWINDOWHWND
- BKM\_QUERYSTATUSLINETEXT
- BKM\_QUERYTABBITMAP
- BKM\_QUERYTABTEXT

## Default Processing

The default message procedure sets *rc* to TRUE.

---

## BKM\_QUERYPAGESTYLE

This message queries the style that was set when the specified page was inserted.

## Parameters

**param1**

**ulPageId** (ULONG)

Page ID.

Page identifier of the page from which to query the style setting.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**usPageStyle** (USHORT)

Page style data.

**BOOKERR\_INVALID\_PARAMETERS** An invalid page ID was specified for the *ulPageId* parameter.

Other Page style data.

## Remarks

This style data is set when the page is inserted, which is done by using the **BKM\_INSERTPAGE** message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## BKM\_QUERYPAGEWINDOWHWND

This message queries the application page window handle associated with the specified page.

## Parameters

**param1**

**ulPageId** (ULONG)

Page ID.

Page identifier of the page whose window handle is requested.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**hwndPage** (HWND)

Window handle.

Handle of the application page window associated with the specified page identifier.

**BOOKERR\_INVALID\_PARAMETERS** An invalid page ID was specified for the *ulPageId* parameter.

**NULLHANDLE** No application page window handle is associated for the page specified in the *ulPageId* parameter.

Other Handle of the application page window associated with the specified page identifier.

## Remarks

The application page window handle is set by using the `BKM_SETPAGEWINDOWHWND` message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `NULLHANDLE`.

---

## BKM\_QUERYSTATUSLINETEXT

This message queries the status line text, text size, or both for the specified page.

### Parameters

**param1**

**ulPageld** (ULONG)

Page ID.

Page identifier of the page whose status line text is requested.

**param2**

**pBookText** (PBOOKTEXT)

Pointer to a `BOOKTEXT` data structure. See “`BOOKTEXT`” on page A-23 for definitions of this structure’s fields as they apply to the `BKM_QUERYSTATUSLINETEXT` message.

### Returns

**statusTextLen** (USHORT)

String length.

Length of the status line text string.

`BOOKERR_INVALID_PARAMETERS` An invalid page ID was specified for the *ulPageld* parameter or the structure specified for the *pBookText* parameter is invalid.

0 No text data has been set (`BKM_SETSTATUSLINETEXT`) for the page specified in the *ulPageld* parameter.

Other Length of the returned status line text string.

### Remarks

The size of the status line text string can be queried by specifying 0 for the *textLen* field of the `BOOKTEXT` data structure. In this way, the application can determine the size of the buffer needed to store the status line text string. The null character at the end of the text string is not included in the returned length.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action other than to return 0.

---

## BKM\_QUERYTABBITMAP

This message queries the bit-map handle associated with the specified page.

### Parameters

**param1**

**ulPageld** (ULONG)

Page ID.

Page identifier of the page whose bit-map handle is requested. This should be a page for which a BKA\_MAJOR or BKA\_MINOR attribute has been specified.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**hbm** (HBITMAP)

Bit-map handle.

Handle of the bit map associated with the specified page identifier.

**BOOKERR\_INVALID\_PARAMETERS** An invalid page ID was specified for the *ulPageld* parameter.

**NULLHANDLE** No bit-map handle is associated with the page specified in the *ulPageld* parameter.

**Other** Handle of the bit map associated with the specified page identifier.

### Remarks

The tab bit-map handle is set by using the BKM\_SETTABBITMAP message.

If this message is sent for a page having both major and minor tab attributes, the notebook returns the bit map associated with the major tab.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return NULLHANDLE.



---

## BKM\_QUERYTABTEXT

This message queries the text, text size, or both for the specified page.

### Parameters

#### param1

##### **ulPageId** (ULONG)

Page ID.

Page identifier of the page whose tab text is requested. This should be a page for which a BKA\_MAJOR or BKA\_MINOR attribute has been specified.

#### param2

##### **pBookText** (PBOOKTEXT)

Pointer to a BOOKTEXT data structure.

See "BOOKTEXT" on page A-23 for definitions of this structure's fields as they apply to the BKM\_QUERYTABTEXT message.

### Returns

#### **tabTextLen** (USHORT)

Length of the tab text string.

|                            |  |
|----------------------------|--|
| BOOKERR_INVALID_PARAMETERS | An invalid page ID was specified for the <i>ulPageId</i> parameter or the structure specified for the <i>pBookText</i> parameter is invalid. |
| 0                          | No text data has been set (BKM_SETTABTEXT) for the page specified in the <i>ulPageId</i> parameter.  |
| Other                      | Length of the returned tab text string.  |

### Remarks

The size of the tab text string can be queried by specifying 0 for the *tabTextLen* field in the BOOKTEXT data structure. In this way, the application can determine the size of the buffer needed to store the tab text string. The null character at the end of the text string is not included in the returned length.

If this message is sent for a page having both major and minor tab attributes, the notebook returns the text which is associated with the major tab.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## BKM\_SETDIMENSIONS

This message sets the height and width for the major tabs, minor tabs, or page buttons.

### Parameters

#### param1

**usWidth** (USHORT)  
Width value to set.

**usHeight** (USHORT)  
Height value to set.

#### param2

**usType** (USHORT)  
Notebook region.

Notebook region for which the dimensions are to be set. Valid values are:

- BKA\_MAJORTAB
- BKA\_MINORTAB
- BKA\_PAGEBUTTON.

### Returns

#### rc (BOOL)

Success indicator.

TRUE Dimensions were successfully set.

FALSE Unable to set dimensions. Returned if an invalid value is specified for the *usType* parameter or if the dimensions are invalid.

### Remarks

If either the BKA\_MAJORTAB or BKA\_MINORTAB attribute is specified for the *usType* parameter, the minimum width and height for display is 7 pels to allow space for the tab border and the selection cursor. If the tabs or page buttons are not to be displayed, the height and width can be set to 0.

If the new dimensions cause the notebook size to change, the notebook sends a BKN\_NEWPAGE\_SIZE notification code to the application.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_SETNOTEBOOKCOLORS

This message sets the colors for the major tab text and background, the minor tab text and background, and the notebook page background.

### Parameters

**param1**

**ulColor** (ULONG)

Color value to set.

**param2**

**usBookAttr** (USHORT)

Notebook region.

Notebook region whose color is to be set. Valid values are:

**BKA\_BACKGROUNDPAGECOLOR** or **BKA\_BACKGROUNDPAGECOLORINDEX**

Page background. This color is initially set to **SYSCLR\_PAGEBACKGROUND**.

**BKA\_BACKGROUNDMAJORCOLOR** or  
**BKA\_BACKGROUNDMAJORCOLORINDEX**

Major tab background. This color is initially set to **SYSCLR\_PAGEBACKGROUND**.

**BKA\_BACKGROUNDMINORCOLOR** or **BKA\_BACKGROUNDMINORCOLORINDEX**

Minor tab background. This color is initially set to **SYSCLR\_PAGEBACKGROUND**.

**BKA\_FOREGROUNDMAJORCOLOR** or  
**BKA\_FOREGROUNDMAJORCOLORINDEX**

Major tab text. This color is initially set to **SYSCLR\_WINDOWTEXT**.

**BKA\_FOREGROUNDMINORCOLOR** or **BKA\_FOREGROUNDMINORCOLORINDEX**

Minor tab text. This color is initially set to **SYSCLR\_WINDOWTEXT**.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Colors were successfully set.

**FALSE** Unable to set colors. Returned if an invalid notebook attribute is specified for the *usBookAttr* parameter.

## Remarks

The notebook background, border, selection cursor, and status line text colors are mapped to system presentation attributes. See WM\_PRESPARAMCHANGED (in Notebook Controls) for information about setting the color of these regions.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_SETPAGEDATA

This message sets the 4 bytes of application reserved storage associated with the specified page.

### Parameters

**param1**

**ulPageId** (ULONG)

Page ID.

The page identifier of the page from which to set the 4 bytes of data.

**param2**

**ulPageData** (ULONG)

Page data.

Application-defined page data.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Page data was successfully set.

FALSE Unable to set page data. This value is returned if the page ID specified in the *ulPageId* parameter is invalid.

## Remarks

This data can be queried by using the BKM\_QUERYPAGEDATA message.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_SETPAGEINFO

This message sets the page information associated with notebook page which contains a single message.

### Parameters

**param1**

**ulPageId** (ULONG)

Id of the notebook page whose information is to be set.

**param2**

**pPageInfo** (PPAGEINFO)

Pointer to a notebook page information structure.

### Returns

**returns**

**rc** (BOOL)

Success indicator.

Possible values are described in the following list:

TRUE    Message was processed.  
FALSE   Message was ignored.

### Remarks

This message provides an application with the ability to associate a window handle, a static dialog resource or a dynamic dialog resource with a notebook page. The notebook can automatically load the dialog resource when the resource is associated with the page or when the page is turned.

This message performs the tasks of the following notebook messages:

- BKM\_SETPAGEDATA
- BKM\_SETPAGEWINDOWHWND
- BKM\_SETSTATUSLINETEXT
- BKM\_SETTABBITMAP
- BKM\_SETTABTEXT

### Default Processing

The default message procedure sets *rc* to TRUE.

---

## BKM\_SETPAGEWINDOWHWND

This message associates an application page window handle with the specified notebook page.

### Parameters

#### param1

##### **ulPageId** (ULONG)

Page ID.

The page ID of the notebook page with which the application page window is to be associated.

#### param2

##### **hwndPage** (HWND)

Window handle.

The handle of the application page window that is to be associated with the notebook page identified in the *ulPageId* parameter.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE Application page window handle was successfully set.

FALSE Unable to set application page window handle. This value is returned if the page ID specified for the *ulPageId* parameter is invalid.

### Remarks

The notebook shows the application page window specified in the *hwndPage* parameter whenever the notebook page specified in the *ulPageId* parameter is brought to the top of the notebook. If the *BKA\_AUTOPAGESIZE* attribute is specified when that page is inserted into the notebook, the notebook also handles the sizing and positioning of the application page window.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_SETSTATUSLINETEXT

This message associates a text string with the specified page's status line.

### Parameters

#### param1

**ulPageld** (ULONG)

Page ID.

The page identifier with which to associate the text string.

#### param2

**pString** (PSZ)

Pointer to a text string that ends in a null character.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Status line text was successfully set.

FALSE Unable to set status line text. This value is returned if the page ID specified in the *ulPageld* parameter is invalid or if the page was inserted without specifying the BKA\_STATUSTEXTON attribute.

### Remarks

If the text is longer than the status area length, only the text that fits in the status area is displayed.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## BKM\_SETTABBITMAP

This message associates a bit-map handle with the specified page.

### Parameters

#### param1

**ulPageld** (ULONG)

Page ID.

The page identifier with which to associate the bit-map handle. This should be a page for which a BKA\_MAJOR or BKA\_MINOR attribute has been specified.

## param2

**hbm** (HBITMAP)  
Bit-map handle.

## Returns

**rc** (BOOL)

Success indicator.

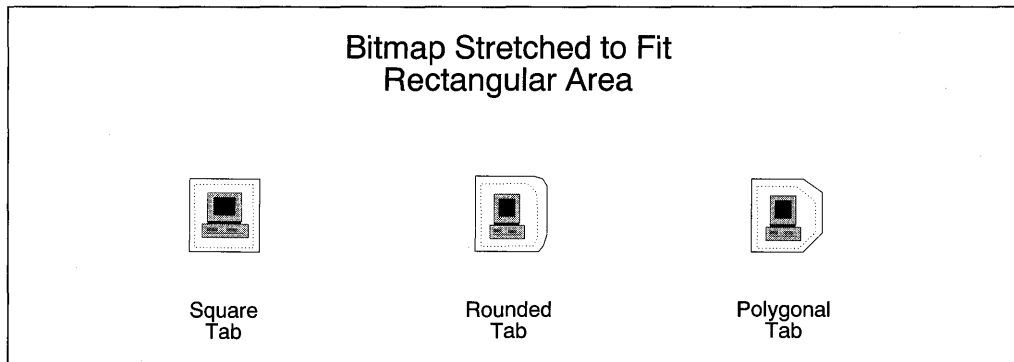
**TRUE** Tab bit map was successfully set.

**FALSE** Unable to set tab bit map. If the page ID specified in the *ulPageId* parameter is invalid or if it identifies a page that does not have a BKA\_MAJOR or BKA\_MINOR attribute, FALSE is returned and no bit map is associated with the page.

## Remarks

If this message is sent for a page having both major and minor tab attributes, the notebook sets both the major and minor tab bit maps.

When displayed, the bit map is stretched to fit the size of the tab. If a tab has rounded or polygonal edges, the bit map is sized to fit the rectangular area of the tab, as shown in Figure 23-1.



*Figure 23-1. Tabs Showing Rectangular Area Used to Size a Bit Map*

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.



---

## BKM\_SETTABTEXT

This message associates a text string with the specified page.

### Parameters

**param1**

**ulPageId** (ULONG)

Page ID.

The page identifier with which to associate the text string. This should be a page for which a BKA\_MAJOR or BKA\_MINOR attribute has been specified.

**param2**

**pString** (PSZ)

Pointer to a text string that ends with a null character.

### Returns

**rc** (BOOL)

Success indicator.

TRUE Tab text was successfully set.

FALSE Unable to set tab text. If the page ID specified in the *ulPageId* parameter is invalid or if it identifies a page that does not have a BKA\_MAJOR or BKA\_MINOR attribute, FALSE is returned and no text string is associated with the page.

### Remarks

The text is centered from the tab edges.

The application can define a mnemonic key when sending this message by placing a tilde (~) character before the character that is to be the mnemonic key. The notebook brings this page to the top whenever the user presses the mnemonic key.

The mnemonic key processing is not case-sensitive, so the user can type the mnemonic character in either upper or lower case.

The application can remove or change the mnemonic key by sending additional BKM\_SETTABTEXT messages for the specified page.

If this message is sent for a page having both major and minor tab attributes, the notebook sets both the major and minor tab text.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## **BKM\_TURNTOPAGE**

This message brings the specified page to the top of the notebook.

### **Parameters**

#### **param1**

**ulPageId** (ULONG)

Page ID.

The page identifier that is to become the top page.

#### **param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Returns**

**fSuccess** (BOOL)

Success indicator.

**TRUE** The page was successfully moved to the top of the notebook.

**FALSE** Unable to move the page to the top of the notebook. This value is returned if the page ID specified in the *ulPageId* parameter is invalid.

### **Remarks**

The application receives a **BKN\_PAGESELECTED** notification code when the new page is brought to the top of the notebook.

### **Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return **FALSE**.

---

## **WM\_CHAR (in Notebook Controls)**

For the cause of this message, see WM\_CHAR.

For a description of the parameters, see WM\_CHAR.

### **Remarks**

If the application page window has the focus (for example, the cursor is on a control within the top page dialog), the notebook handles the following keyboard interaction:

**Alt+Up Arrow** Sets the focus to the notebook window.

If the notebook control has the focus (for example, the cursor is on the major tab, minor tab or page turning button), the notebook handles the following keyboard interactions:

**Alt+Down Arrow** Sets the focus to the application page window.

**Tab** Moves the selection cursor to the next position or control.

**Shift+Tab** Moves the selection cursor to the previous position or control.

### **Down Arrow or Right Arrow**

Moves the selection cursor to the next major or minor tab. If either of these keys is pressed while the selection cursor is on a major tab, the cursor moves to the next major tab. If either of these keys is pressed while the selection cursor is on a minor tab, the cursor moves to the next minor tab. If the next tab is not visible, the tabs are scrolled to bring the next tab into view. If the end of the tabs is reached, scrolling ends.

### **Up Arrow or Left Arrow**

Moves the selection cursor to the previous major or minor tab. If either of these keys is pressed while the selection cursor is on a major tab, the cursor moves to the previous major tab. If either of these keys is pressed while the selection cursor is on a minor tab, the cursor moves to the previous minor tab. If the previous tab is not visible, the tabs are scrolled to bring the previous tab into view. If the beginning of the tabs is reached, scrolling ends.

### **Enter or Spacebar**

The cursored tab page becomes the top page of the notebook.

### **Mnemonics**

Brings the page whose tab contains the mnemonic character to the top of the notebook whenever the user presses the mnemonic key. Mnemonic key definition is provided by using the BKM\_SETTABTEXT message. Coding a mnemonic character (   ) before a text character in the BKM\_SETTABTEXT message causes that character to be underlined in the tab's text string and activates it as a mnemonic selection character. The mnemonic key pressing is not case-sensitive, so the user can type the mnemonic character in either upper or lower case.

### **PgDn or Alt+PgDn**

Brings the next page to the top of the notebook and sets the selection cursor on the associated tab, if there is one.

|                         |   |
|-------------------------|---|
| <b>PgUp or Alt+PgUp</b> | Brings the previous page to the top of the notebook and sets the selection cursor on the associated tab, if there is one. |
| <b>Home</b>             | Brings the first page of the notebook to the top and sets the selection cursor on the associated tab, if there is one.    |
| <b>End</b>              | Brings the last page of the notebook to the top and sets the selection cursor on the associated tab, if there is one.     |

### Default Processing

For a description of the default processing, see WM\_CHAR.

## WM\_PRESPARAMCHANGED (in Notebook Controls)

For the cause of this message, see WM\_PRESPARAMCHANGED.

### Parameters

#### param1

##### attrtype (ULONG)

Attribute type.

Presentation parameter attribute identity.

##### PP\_BACKGROUNDCOLOR or PP\_BACKGROUNDCOLORINDEX

Sets the background color of the notebook window. This color is initially set to SYSCLR\_FIELDBACKGROUND.

##### PP\_BORDERCOLOR or PP\_BORDERCOLORINDEX

Sets the color of the notebook outline. This color is initially set to SYSCLR\_WINDOWFRAME.

##### PP\_FOREGROUNDCOLOR or PP\_FOREGROUNDCOLORINDEX

Sets the color of text on the status line. This color is initially set to SYSCLR\_WINDOWTEXT.

##### PP\_HILITEBACKGROUNDCOLOR or PP\_HILITEBACKGROUNDCOLORINDEX

Sets the color of the selection cursor. This color is initially set to SYSCLR\_HILITEBACKGROUND.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

**Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

**Remarks**

The application uses this message to notify the notebook that a given inherited presentation parameter has changed.

**Default Processing**

For a description of the default processing, see WM\_PRESPARAMCHANGED.

---

**WM\_SIZE (in Notebook Controls)**

For the cause of this message, see WM\_SIZE.

For a description of the parameters, see WM\_SIZE.

**Remarks**

When the size of the notebook window changes, all of the regions are recalculated. The notebook sends a BKN\_NEWPAGESIZE notification code to the application. The notebook sets the position and size of application page windows that are associated with pages for whom the BKA\_AUTOPAGESIZE attribute is set.

**Default Processing**

For a description of the default processing, see WM\_SIZE.

---

## Chapter 24. Slider Control Window Processing

This system-provided window procedure processes the actions on a slider control (WC\_SLIDER).

---

### Purpose

A slider control (WC\_SLIDER window class) is a visual component whose specific purpose is to allow a user to set, display, or modify a value by moving a slider arm along a slider shaft. Sliders are typically used to allow a user to easily set values that have familiar increments, such as feet, inches, degrees, decibels, and so forth.

However, they can also be used for other purposes when immediate feedback is necessary, such as to blend colors or to show the percentage of a task that has completed. For example, an application might allow a user to mix and match color shades by moving a slider arm, or a read-only slider could be provided that shows how much of a task has completed by filling in the slider shaft as the task progresses. These are just a few examples to show you the many ways in which sliders can be used.

The appearance of and user interaction for a slider is similar to the appearance of and user interaction for a scroll bar. However, these two controls are not interchangeable because each has a distinct purpose. The scroll bar is used to scroll into view information that is outside a window's client area, while the slider is used to set, display, or modify that information, whether it is in the client area or not in the client area.

The slider is designed to be customizable to meet varying application requirements, while providing an easy-to-use user interface component that can be used to develop products that conform to the Common User Access (CUA) user interface guidelines. The application can specify different scales, sizes, and orientations for its sliders, but the underlying function of the control remains the same. For a complete description of CUA sliders, refer to the *SAA CUA Guide to User Interface Design* and the *SAA CUA Advanced Interface Design Reference*.

---

### Slider Control Styles

Slider control window styles are set when a slider window is created. The following styles can be set when creating a slider control window. If no styles are specified, defaults, which are identified in the following descriptions, are used.

- Specify either of the following to determine the slider's orientation:

#### **SLS\_HORIZONTAL**

The slider is positioned horizontally. The slider arm can move left and right on the slider shaft. A scale can be placed on top of the slider shaft, below the slider shaft, or in both places. This is the default orientation of the slider.

**SLS\_VERTICAL**

The slider is positioned vertically. The slider arm can move up and down the slider shaft. A scale can be placed on the left side of the slider shaft, on the right side of the slider shaft, or in both places.

- Specify one of the following to position the slider within the slider window:

**SLS\_CENTER**

The slider is centered in the slider window. This is the default positioning of the slider.

**SLS\_BOTTOM**

The slider is positioned at the bottom of the slider window. This is valid for horizontal sliders only.

**SLS\_TOP**

The slider is positioned at the top of the slider window. This is valid for horizontal sliders only.

**SLS\_LEFT**

The slider is positioned at the left edge of the slider window. This is valid for vertical sliders only.

**SLS\_RIGHT**

The slider is positioned at the right edge of the slider window. This is valid for vertical sliders only.

- Specify one of the following to determine the location of the scale on the slider shaft:

**SLS\_PRIMARYSCALE1**

The slider uses the increment and spacing specified for scale 1 as the incremental value for positioning the slider arm. Scale 1 is displayed above the slider shaft of a horizontal slider and to the right of the slider shaft of a vertical slider. This is the default for a slider.

**SLS\_PRIMARYSCALE2**

The slider uses the increment and spacing specified for scale 2 as the incremental value for positioning the slider arm. Scale 2 is displayed below the slider shaft of a horizontal slider and to the left of the slider shaft of a vertical slider.

- Specify one of the following to determine the slider arm's home position:

**SLS\_HOMELEFT**

The slider uses the left edge of the slider as the base value for incrementing. This is the default for horizontal sliders and is valid for horizontal sliders only.

**SLS\_HOMERIGHT**

The slider uses the right edge of the slider as the base value for incrementing. This is valid for horizontal sliders only.

**SLS\_HOMEBOTTOM**

The slider uses the bottom of the slider as the base value for incrementing. This is the default for vertical sliders and is valid for vertical sliders only.

**SLS\_HOMETOP**

The slider uses the top of the slider as the base value for incrementing. This is valid for vertical sliders only.

- Specify one of the following to determine the location of the slider buttons. If you do not specify one of these styles, or if conflicting styles are specified, slider buttons are not included in the slider control.

**SLS\_BUTTONSLEFT**

The slider includes incremental slider buttons with the control and places them to the left of the slider shaft. These slider buttons move the slider arm by one position, either left or right, in the direction that is selected. This is valid for horizontal sliders only.

**SLS\_BUTTONSRIGHT**

The slider includes incremental slider buttons with the control and places them to the right of the slider shaft. These slider buttons move the slider arm by one position, either left or right, in the direction that is selected. This is valid for horizontal sliders only.

**SLS\_BUTTONSBOTTOM**

The slider includes incremental slider buttons with the control and places them at the bottom of the slider shaft. These slider buttons move the slider arm by one position, either up or down, in the direction that is selected. This is valid for vertical sliders only.

**SLS\_BUTTONSTOP**

The slider includes incremental slider buttons with the control and places them at the top of the slider shaft. These slider buttons move the slider arm by one position, either up or down, in the direction that is selected. This is valid for vertical sliders only.

- Other styles that you can specify:

**SLS\_SNAPTOINCREMENT**

The slider arm, when moved to a position between two specified values on the slider scale, such as between two tick marks, is positioned on the nearest value and is redrawn at that position. If this style is not specified, the slider arm remains at the position to which it is moved.

**SLS\_READONLY**

The slider is created as a read-only slider. This means that the user cannot interact with the slider. It is used merely as a mechanism to present a quantity to the user, such as the percentage of completion of an ongoing task. Visual differences for a read-only slider include a narrow slider arm, no slider buttons and no detents.



**SLS\_RIBBONSTRIP**

As the slider arm moves, the slider fills the slider shaft between the home position and the slider arm with a color value that is different from the slider shaft color, similar to the mercury in a thermometer.

**SLS\_OWNERDRAW**

The application is notified whenever the slider shaft, the ribbon strip, the slider arm, and the slider background are to be drawn.

---

**Slider Control Data**

See "SLDCDATA" on page A-187.

---

## Slider Control Notification Messages

These messages are initiated by the slider control window to notify its owner of significant events.

---

### WM\_CONTROL (in Slider Controls)

For the cause of this message, see WM\_CONTROL.

#### Parameters

##### param1

**id** (USHORT)

Slider control identity.

**notifycode** (USHORT)

Notification code.

The slider control uses these notification codes:

|                 |   |
|-----------------|---|
| SLN_CHANGE      | The slider arm position has changed.                        |
| SLN_KILLFOCUS   | The slider control is losing the focus.                     |
| SLN_SETFOCUS    | The slider control is receiving the focus.                  |
| SLN_SLIDERTRACK | The slider arm is being dragged, but has not been released. |

##### param2

**notifyinfo** (ULONG)

Control-specific information.

When the value of the *notifycode* parameter is SLN\_CHANGE or SLN\_SLIDERTRACK, this value is the new arm position, expressed as the number of pixels from the home position.

Otherwise, this value is the window handle (HWND) of the slider control.

#### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

#### Remarks

The slider control window procedure generates this message and sends it to its owner, informing the owner of this event.

#### Default Processing

For a description of the default processing, see WM\_CONTROL.

---

## WM\_CONTROLPOINTER (in Slider Controls)

For the cause of this message, see WM\_CONTROLPOINTER.

For a description of the parameters, see WM\_CONTROLPOINTER.

### Remarks

For the appropriate remarks, see WM\_CONTROLPOINTER.

### Default Processing

For the default processing, see WM\_CONTROLPOINTER.

---

## WM\_DRAWITEM (in Slider Controls)

If the SLS\_OWNERDRAW style bit is set for a slider control, this notification message is sent to that slider control's owner whenever the slider shaft, ribbon strip, slider arm, and slider background are to be drawn.

### Parameters

**param1**

**id** (USHORT)

Window identifier.

The window identifier of the slider control sending this notification message.

**param2**

**powneritem** (OWNERITEM)

Pointer to an OWNERITEM data structure.

The following list defines the OWNERITEM data structure fields that apply to the slider control. See OWNERITEM for the default field values.

**hwnd** (HWND)

Slider window handle.

**hps** (HPS)

Presentation-space handle.

**fsState** (ULONG)

Slider window style flags. See "Slider Control Styles" on page 24-1 for descriptions of these style flags.

**fsAttribute** (ULONG)

Reserved.

**fsStateOld** (ULONG)

Reserved.

*fsAttributeOld* (ULONG)

Reserved.

*rollItem* (RECTL)

Item rectangle to be drawn in window coordinates.

*idItem* (LONG)

Identity of item to be drawn:

SDA\_SLIDERSHAFT Specifies that the slider shaft is to be drawn.

SDA\_RIBBONSTRIP Specifies that the slider shaft area that contains a ribbon strip is to be drawn.

SDA\_SLIDERARM Specifies that the slider arm is to be drawn.

SDA\_BACKGROUND Specifies that the slider background is to be drawn.

*hItem* (ULONG)

Reserved.

## Returns

*rc* (BOOL)

Item-drawn indicator.

TRUE The owner draws the item.

FALSE If the owner does not draw the item, the owner returns this value and the slider control draws the item.

## Remarks

The slider control provides this message to give the application the opportunity to provide a custom slider shaft, custom ribbon strip, custom slider arm, and custom background. The application can specify one or all of these items and is given the opportunity to do so.

The slider control window procedure generates this message and sends it to its owner, informing the owner that an item is to be drawn. The owner is then given the opportunity to draw that item, and to indicate that an item has been drawn or that the slider control is to draw it.

## Default Processing

For a description of the default processing, see WM\_DRAWITEM.

---

## Slider Control Window Messages

This section describes the slider control window procedure actions on receiving the following messages.

---

### SLM\_ADDDETENT

This message places a detent along the slider shaft at the position specified on the primary scale. A detent is an indicator that represents a predefined value for a quantity. It does not have to correspond to an increment of the slider.

#### Parameters

##### param1

###### **usDetentPos** (USHORT)

Detent position.

Number of pixels the detent is positioned from home.

##### param2

###### **ulReserved** (ULONG)

Reserved value, should be 0.

#### Returns

##### **ulDetentId** (ULONG)

Detent ID.

Unique identifier for the detent being added to the slider. If 0 is returned, an error occurred. The `WinGetLastError` function may return the following errors:

- `PMERR_HEAP_MAX_SIZE_REACHED`
- `PMERR_PARAMETER_OUT_OF_RANGE`.

#### Remarks

The application uses this message to add detents along the slider to denote values that do not fall along an increment setting. An example of this would be a slider that represents temperature and has increments that are on multiples of 5. A detent could be located at 32, instead of 30 or 35, for special purposes.

#### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_QUERYDETENTPOS

This message queries for the current position of a detent.

### Parameters

**param1**

**ulDetentId** (ULONG)

Detent ID.

Unique detent identifier, which indicates the position to be returned.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ReturnCode**

**usDetentPos** (USHORT)

Detent position.

Number of pixels the detent is positioned from home.

>= 0

Number of pixels the detent is positioned from home.

SLDERR\_INVALID\_PARAMETERS

An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

**fDetentLocation** (USHORT)

Scale.

The scale along which the detent is located. One of the following:

SMA\_SCALE1 Detent position is along scale 1.

SMA\_SCALE2 Detent position is along scale 2.

### Remarks

An application could use this message to place text above the detent or position an item relative to it.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_QUERYSCALETEXT

This message queries for the text associated with a tick mark for the primary scale and copies that text into a buffer.

### Parameters

#### param1

##### **usTickNum** (USHORT)

Tick location.

Tick location to query for the text.

##### **usBufLen** (USHORT)

Buffer length.

Length of the buffer to copy the text into. The buffer size should include space for the null termination character.

#### param2

##### **pTickText** (PSZ)

Pointer to the buffer into which to place the text string for the tick mark.

### Returns

#### **sTextLen** (SHORT)

Count of bytes.

Count of bytes copied to buffer.

$\geq 0$

Length of the text string, excluding the null termination character.

#### SLDERR\_INVALID\_PARAMETERS

An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

### Remarks

This message could be used to return text that represents the current position of the slider arm or to query the text for use in ownerdraw mode.

By specifying 0 as the value of the *usBufLen* parameter and then looking at the value returned in the *sTextLen* parameter, an application can determine the size of the buffer to allocate for copying the text. An application can then allocate a buffer of this size, adding one byte for the null termination character, and then specify this buffer and size on the query call.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_QUERYSLIDERINFO

This message queries the current position or dimensions of a key component of the slider. The information returned and its format depends on the type of information requested.

### Parameters

#### param1

##### usInfoType (USHORT)

Information attribute.

Attribute that identifies the requested information. It can be one of the following:

|                         |  |
|-------------------------|--|
| SMA_SHAFTDIMENSIONS     | Queries for the length and breadth of the slider shaft.  |
| SMA_SHAFTPOSITION       | Queries for the x-, y-position of the lower-left corner of the slider shaft.   |
| SMA_SLIDERARMDIMENSIONS | Queries for the length and breadth of the slider arm.  |
| SMA_SLIDERARMPOSITION   | Queries for the position of the slider arm. The position can be returned either as an increment position or a range value. |

##### usArmPosType (USHORT)

Format attribute.

Attribute that identifies the format in which the information should be returned if the slider arm position is requested. This value is ignored for all other queries and is one of the following:

|                    |  |
|--------------------|--|
| SMA_RANGEVALUE     | The value returned represents the number of pixels between the home position and the current arm position in the low order byte. The high order byte represents the pixel count of the entire range of the slider control. |
| SMA_INCREMENTVALUE | The value returned represents an increment position using the primary scale.   |

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.



## Returns

### ullInfo (ULONG)

Return information.

One of the following items, depending on which SMA\_\* message attribute or attributes, were set with the SLM\_SETSLIDERINFO message:

- If the SMA\_SHAFTDIMENSIONS attribute is set, the following is returned:
  - usShaftLength* (USHORT)  
Length of the slider shaft, in pixels. It is the width of the slider shaft for horizontal sliders, and the height of the slider shaft for vertical sliders.
  - usShaftBreadth* (USHORT)  
Breadth of the slider shaft, in pixels. It is the height of the slider shaft for horizontal sliders, and the width of the slider shaft for vertical sliders.
- If the SMA\_SHAFTPOSITION attribute is set, the following is returned:
  - xShaftCoord* (USHORT)  
X-coordinate of the slider shaft position within the slider window. This value is expressed in window coordinates and represents the lower-left corner of the slider shaft.
  - yShaftCoord* (USHORT)  
Y-coordinate of the slider shaft position within the slider window. This value is expressed in window coordinates and represents the lower-left corner of the slider shaft.
- If the SMA\_SLIDERARMDIMENSIONS attribute is set, the following is returned:
  - usArmLength* (USHORT)  
Length of the slider arm, in pixels. It is the width of the slider arm for horizontal sliders and the height of the slider arm for vertical sliders.
  - usArmBreadth* (USHORT)  
Breadth of the slider arm, in pixels. It is the height of the slider arm for horizontal sliders and the width of the slider arm for vertical sliders.
- If the SMA\_SLIDERARMPOSITION and SMA\_INCREMENTVALUE attributes are set, the following is returned:
  - usArmPos* (USHORT)  
Number of pixels from the home position to the slider arm.
  - usSliderRange* (USHORT)  
Number of pixels over which the user could select a value on the slider.
- If the SMA\_SLIDERARMPOSITION and SMA\_INCREMENTVALUE attributes are set, the following is returned:
  - usIncrementPos* (USHORT)  
Increment that corresponds to the current position of the slider arm.

- If the SLDERR\_INVALID\_PARAMETERS error is returned, an error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Remarks

The application uses this message to query for information about individual parts of a slider control, or the value selected by a user.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_QUERYTICKPOS

This message queries for the current position of a tick mark for the primary scale. This represents where the tick mark would be located. The tick mark does not have to have a size (that is, to be visible) to use this message.

### Parameters

**param1**

**usTickNum** (USHORT)

Tick mark location.

Specifies the tick mark location to query for the position.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ReturnCode**

**xTickPos** (USHORT)

X-coordinate.

X-coordinate of the point that represents the position of the tick mark. It is the starting position of the tick mark and represents the end of the tick mark closest to the slider shaft.

**yTickPos** (USHORT)

Y-coordinate.

Y-coordinate of the point that represents the position of the tick mark. It is the starting position of the tick mark and represents the end of the tick mark closest to the slider shaft.

If NULL is returned in either parameter, an error occurred. The WinGetLastError function may return the following error:

PMERR\_PARAMETER\_OUT\_OF\_RANGE.

### Remarks

This message could be used to get the position of a tick mark along the slider for use in ownerdraw mode if, for example, you want to place something other than text, such as bit maps or icons, above the tick marks.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_QUERYTICKSIZE

This message queries for the size of a tick mark for the primary scale. All tick marks default to a size of 0 (invisible) if not set by the application with the SLM\_SETTICKSIZE message.

### Parameters

**param1**

**usTickNum** (USHORT)

Tick mark location.

Specifies the tick mark location to query for the size.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usTickSize** (USHORT)

Tick mark length.

Specifies the length of the tick mark at the position queried, in pixels. If this value is 0, the tick mark is invisible.

If the SLDERR\_INVALID\_PARAMETERS error is returned, an error occurred. The WinGetLastError function may return the following error:

PMERR\_PARAMETER\_OUT\_OF\_RANGE.

### Remarks

The application uses this message to query a scale along the slider to indicate what tick marks, tick mark sizes, or both are currently set for the slider.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## SLM\_REMOVEDETENT

This message removes a previously specified detent. A detent is an indicator that represents a predefined value for a quantity and does not have to correspond to an increment of the slider.

### Parameters

**param1**

**ulDetentId** (ULONG)

Detent ID.

Unique detent identifier for the detent that is to be removed from the slider.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Detent was successfully removed.

**FALSE** An error occurred. The WinGetLastError function may return the following error:

PMERR\_INVALID\_PARAMETERS.

### Remarks

The application uses this message to remove detents added previously to the slider to denote values that do not fall along an increment setting.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## SLM\_SETSCALETEXT

This message sets text above a tick mark for the primary scale. A tick mark does not have to be visible to have text set above it. The text is centered on the tick mark.

### Parameters

#### param1

**usTickNum** (USHORT)

Tick mark location.

Specifies the tick mark location that is to have the text placed with it.

#### param2

**pTickText** (PSZ)

Pointer to the text that is to be drawn at the position specified.

If this value is NULL, no text is drawn.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** Text was successfully added to the scale.

**FALSE** An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_HEAP\_MAX\_SIZE\_REACHED
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

### Remarks

The application uses this message to draw text along the increments of the slider to clarify the magnitude of the range. This text could show the exact value for that tick mark, or could be a general remark, such as low, high, and so forth.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## SLM\_SETSLIDERINFO

This message sets the current position or dimensions of a key component of the slider. The component to be changed is indicated by one parameter and the new value is placed in the other.

### Parameters

**param1**

#### **usInfoType** (USHORT)

Component attribute.

Identifies the slider component that is to be modified. Specify one of the following:

|                         |  |
|-------------------------|--|
| SMA_SHAFTDIMENSIONS     | Sets the width (for vertical sliders) or height (for horizontal sliders) of the slider shaft.                      |
| SMA_SHAFTPOSITION       | Sets the x-, y-position of the lower-left corner of the slider shaft in the slider window.                         |
| SMA_SLIDERARMDIMENSIONS | Sets the width and height of the slider arm.   |
| SMA_SLIDERARMPOSITION   | Sets the position of the slider arm. This value can be specified either as an increment position or a range value. |

#### **usArmPosType** (USHORT)

Format attribute.

Identifies the format in which the information should be interpreted by the slider if setting the slider arm position is requested. This value is a reserved field for other set requests. The format is one of the following:

|                    |  |
|--------------------|--|
| SMA_RANGEVALUE     | Number of pixels between the home position and the current arm position. |
| SMA_INCREMENTVALUE | Increment position using the primary scale.                              |

## param2

### **ullInfo** (ULONG)

New value.

New value to change the slider component to. The format of the information depends on the component being changed and is indicated by the SMA\_\* message attribute or attributes that are set.

- If the SMA\_SHAFTDIMENSIONS attribute is set, the *ullInfo* parameter is as follows:

#### *usShaftBreadth* (USHORT)

Width (for vertical sliders) or height (for horizontal sliders) the slider shaft should be set to, in pixels. This is the breadth the shaft should be.

- If the SMA\_SHAFTPOSITION attribute is set, the *ullInfo* parameter is as follows:

#### *xShaftCoord* (USHORT)

X-coordinate to set the position of the shaft to within the slider window.

This value is expressed in window coordinates and represents the lower-left corner of the shaft.

#### *yShaftCoord* (USHORT)

Y-coordinate to set the position of the shaft to within the slider window.

This value is expressed in window coordinates and represents the lower-left corner of the shaft.

- If the SMA\_SLIDERARMDIMENSIONS attribute is set, the *ullInfo* parameter is as follows:

#### *usArmLength* (USHORT)

Length of the slider arm, in pixels. This is the width of the arm for horizontal sliders and the height of the arm for vertical sliders.

#### *usArmBreadth* (USHORT)

Breadth of the slider arm, in pixels. This is the height of the arm for horizontal sliders and the width of the arm for vertical sliders.

- If the SMA\_SLIDERARMPOSITION and SMA\_RANGEVALUE attributes are set, the *ullInfo* parameter is as follows:

#### *usArmPos* (USHORT)

Number of pixels to be set from home to the slider arm.

- If the SMA\_SLIDERARMPOSITION and SMA\_INCREMENTVALUE attributes are set, the *ullInfo* parameter is as follows:

#### *usIncrementPos* (USHORT)

Increment value which corresponds to the position the slider arm should be set to.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Slider component was successfully set.

**FALSE** An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

## Remarks

The application uses this message to customize the slider for a specific use. In setting the shaft dimensions, only the breadth of the slider can be set. The length of the shaft is always determined by the number of increments and the spacing between increments, both of which are set for the primary scale when the slider is created.

Positioning of the shaft within the slider window could be used by applications that cannot use the default positioning provided by the slider control.

Setting of the slider arm dimensions could be used by applications that need a larger slider arm, such as touch screen applications.

Setting the slider arm position can be used to:

- Set the initial value of the slider before it becomes visible.
- Change the value when it is tied to another control, such as an entry field.
- Show the value of a quantity when the slider is being used to monitor an event, such as a read-only slider being used as a progress indicator.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## SLM\_SETTICKSIZE

This message sets the size of a tick mark for the primary scale. All tick marks are initially set to a size of 0 (invisible). Each tick mark along a scale can be set to the size desired.

## Parameters

**param1**

**usTickNum** (USHORT)

Tick mark location.

Tick mark location whose size is to be changed. If the SMA\_SETALLTICKS attribute is specified for this parameter, all tick marks on the primary scale are set to the size specified.



**usTickSize** (USHORT)

Tick mark length.

Length of the tick mark, in pixels. If set to 0, the tick mark will not be drawn.

**param2****ulReserved** (ULONG)

Reserved value, should be 0.

**Returns****rc** (BOOL)

Success indicator.

**TRUE** Tick mark position was successfully set.

**FALSE** An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_HEAP\_MAX\_SIZE\_REACHED
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

**Remarks**

The application uses this message to draw a scale along the slider to indicate value positions in relation to the slider arm. The application can set varying lengths for different increments of the slider to help the user understand the magnitude of the value being set.

**Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

**WM\_CHAR (in Slider Controls)**

For the cause of this message, see WM\_CHAR.

For a description of the parameters, see WM\_CHAR.

**Remarks**

The slider control window procedure responds to this message by sending it to its owner if it has not processed the key stroke. This is the most common means by which the input focus is switched around the various controls in a dialog box.

The keystrokes processed by a linear slider control are:

**Down Arrow** Moves the slider arm down one increment. When the slider arm reaches the bottom of the slider shaft or when a horizontal slider is being used, the Down Arrow key has no effect.

|                          |   |
|--------------------------|---|
| <b>Up Arrow</b>          | Moves the slider arm up one increment. When the slider arm reaches the top of the slider shaft or when a horizontal slider is being used, the Up Arrow key has no effect.   |
| <b>Left Arrow</b>        | Moves the slider arm left one increment. When the slider arm reaches the leftmost edge or when a vertical slider is being used, the Left Arrow key has no effect.   |
| <b>Right Arrow</b>       | Moves the slider arm right one increment. When the slider arm reaches the rightmost edge or when a vertical slider is being used, the Right Arrow key has no effect.  |
| <b>Shift+Down Arrow</b>  | Moves the slider arm to the next detent below the current position. If there are no more detents or if a horizontal slider is being used, the Shift+Down Arrow key combination has no effect.   |
| <b>Shift+Up Arrow</b>    | Moves the slider arm to the next detent above the current position. If there are no more detents or if a horizontal slider is being used, the Shift+Up Arrow key combination has no effect.   |
| <b>Shift+Left Arrow</b>  | Moves the slider arm to the next detent left of the current position. If there are no more detents or if a vertical slider is being used, the Shift+Left Arrow key combination has no effect.   |
| <b>Shift+Right Arrow</b> | Moves the slider arm to the next detent right of the current position. If there are no more detents or if a vertical slider is being used, the Shift+Right Arrow key combination has no effect.   |
| <b>Home, Ctrl+Home</b>   | Moves the slider arm to the home position of the slider. Pressing the Home key or the Ctrl+Home key combination when the slider arm is at the home position has no effect. The default home position for a slider is the leftmost edge for horizontal sliders and the bottom edge for vertical sliders. |
| <b>End, Ctrl+End</b>     | Moves the slider arm to the end position of the slider. Pressing the End key or the Ctrl+End key combination when the slider arm is at the end position has no effect. The default end position for a slider is the rightmost edge for horizontal sliders and the top edge for vertical sliders.        |

A circular slider control only processes left and right arrow keystrokes. These keys move the slider arm one increment to the left or right.

### **Default Processing**

For a description of the default processing, see WM\_CHAR.

---

## **WM\_PRESPARAMCHANGED (in Slider Controls)**

For the cause of this message, see WM\_PRESPARAMCHANGED.

### **Parameters**

#### **param1**

##### **attrtype (ULONG)**

Attribute type.

Presentation parameter attribute identity. The following presentation parameters are initialized by the slider control. The initial value of each is shown in the following list:

##### **PP\_FOREGROUND\_COLOR or PP\_FOREGROUND\_COLORINDEX**

Item foreground color; used when displaying text and bit maps. This color is initialized to SYSCLR\_WINDOWTEXT.

##### **PP\_BACKGROUND\_COLOR or PP\_BACKGROUND\_COLORINDEX**

Slider background color; used for entire control as the background. This color is initialized to SYSCLR\_WINDOW.

#### **param2**

##### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

##### **ulReserved (ULONG)**

Reserved value, must be 0.

### **Remarks**

The application uses this message to notify the slider that a given inherited presentation parameter has changed.

### **Default Processing**

For a description of the default processing, see WM\_PRESPARAMCHANGED.

---

## WM\_QUERYWINDOWPARAMS (in Slider Controls)

For the cause of this message, see WM\_QUERYWINDOWPARAMS.

### Parameters

#### param1

##### **pwndparams** (PWNDPARAMS)

Pointer to a WNDPARAMS window parameter structure.

This structure contains:

##### *status* (USHORT)

Window parameter selection.

Identifies the window parameters that are to be set or queried. Valid values for the slider control are:

**WPM\_CBCTLDATA** Window control data length.

**WPM\_CTLDATA** Window control data.

The flags in the *status* field are cleared as each item is processed. If the call is successful, the *status* field is 0. If any item has not been processed, the flag for that item is still set.

##### *length* (USHORT)

Length of the window text.

##### *text* (PSZ)

Window text.

##### *presparamslength* (USHORT)

Length of presentation parameters.

##### *presparams* (PVOID)

Presentation parameters.

##### *ctldatalength* (USHORT)

Length of window class-specific data.

##### *ctldata* (PVOID)

Window class-specific data.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

## Remarks

The slider control window procedure responds to this message by returning the information in the buffer provided. If this message is sent to a slider window of another process, the information in, or identified by, the value of the *pwdparams* field must be in memory shared by both processes.

## Default Processing

For a description of the default processing, see WM\_QUERYWINDOWPARAMS.

---

## WM\_SETWINDOWPARAMS (in Slider Controls)

For the cause of this message, see WM\_SETWINDOWPARAMS.

## Parameters

**param1**

**pwdparams** (PWNDPARAMS)

Pointer to a WNDPARAMS window parameter structure.

This structure contains:

**status** (USHORT)

Window parameter selection.

Identifies the window parameters that are to be set or queried. The valid value for the slider control is:

**WPM\_CTLDATA** Window control data.

The flags in the *status* field are cleared as each item is processed. If the call is successful, the *status* field is 0. If any item has not been processed, the flag for that item is still set.

**length** (USHORT)

Length of the window text.

**text** (PSZ)

Window text.

**presparamslength** (USHORT)

Length of presentation parameters.

**presparams** (PVOID)

Presentation parameters.

*ctldatalength* (USHORT)

Length of window class-specific data.

*ctldata* (PVOID)

Window class-specific data.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

**Returns**

**rc** (BOOL)

Success indicator.

TRUE Successful operation

FALSE Error occurred.

**Remarks**

If this message is sent to a slider window of another process, the information in, or identified by, the value of the *pwndparams* field must be in memory shared by both processes.

**Default Processing**

For a description of the default processing, see WM\_SETWINDOWPARAMS.



---

## Chapter 25. Circular Slider Control Window Messages

The system-provided window procedure processes the actions on a circular control (WC\_CIRCULARSLIDER).

### Purpose

The circular slider control supports values set in analog rather than digital form. This control is intended to emulate the actual controls of stereo and video components.

The circular slider can be used instead of a linear slider. While, at present, there are no particular guidelines as to when a circular slider should replace a linear slider, the circular slider consumes less space on the screen and, therefore, is practical to represent several controls in the same window. For example, for an audio attributes dialog that has volume, balance, bass, and treble controls, you might want to use a linear slider for the volume control (since it is used frequently); but to conserve space and give a more familiar appearance, the circular slider could be used for the balance, bass, and treble.

---

### Circular Slider Control Styles

These circular slider control styles are available:

|                          |  |
|--------------------------|--|
| <b>CSS_CIRCULARVALUE</b> | Draws a circular thumb, rather than a line, for the value indicator.   |
| <b>CSS_MIDPOINT</b>      | Makes the mid-point tick mark larger.  |
| <b>CSS_NOBUTTON</b>      | Does not display value buttons.  |
| <b>CSS_NONUMBER</b>      | Does not display the value on the dial.  |
| <b>CSS_NOTEXT</b>        | Does not display title text under the dial.  |
| <b>CSS_POINTSELECT</b>   | Permits the values on the circular slider to change immediately when dragged.<br><br>Direct manipulation is performed by using a mouse to click on and drag the circular slider. There are two modes of direct manipulation for the circular slider.<br><br>The default direct manipulation mode is to <i>scroll</i> to the value indicated by the position of the mouse. This could be important if you used a circular slider for a volume control, for example. Increasing the volume from 0% to 100% too quickly could result in damage to both the user's ears and the equipment.<br><br>The other mode of direct manipulation permits the value on the circular slider to change immediately when dragged. This mode is enabled using the <b>CSS_POINTSELECT</b> style bit. When this style is used, |



|                              |   |
|------------------------------|---|
|                              | the value of the dial can be changed by tracking the value with the mouse, which changes values quickly.  |
| <b>CSS_PROPORTIONALTICKS</b> | Allow the length of the tick marks to be calculated as a percentage of the radius.  |
| <b>CSS_360</b>               | Permits the scroll range to extend 360 degrees.<br>CSS_360 forces the CSS_NONNUMBER style on. This is necessary to keep the value indicator from corrupting the number value. |

---

## **Circular Slider Control Data**

See CSBITMAPDATA.

---

## **Default Colors**

The following system colors are used when the system draws button controls:

**SYSCLR\_BACKGROUND**  
**SYSCLR\_FOREGROUND**

Some of these defaults can be replaced by using the following presentation parameters in the application resource script file or source code:

**PP\_BACKGROUND**  
**PP\_BORDER**

---

## Circular Slider Control Notification Messages

These messages are initiated by the circular slider control window to notify its owner of significant events.

---

### WM\_CONTROL (in Circular Slider Controls)

This message occurs when a control has a significant event to notify to its owner.

#### Parameters

**param1**

**usID** (USHORT)

Control-window identity.

The identity of the circular slider that generated the notification.

**usnotifycode** (USHORT)

Notification code.

The notification codes that indicate what action has occurred.

**CSN\_SETFOCUS**

This code returns a Boolean indicating whether the circular slider control sending the notification message is gaining or losing the focus.

*param2* contains TRUE if the control is gaining the focus.

**CSN\_CHANGED**

This code is sent to notify the application that the circular slider value has been changed.

*param2* contains the new value of the circular slider.

**CSN\_TRACKING**

This code is sent to notify the application that the circular slider is being tracked by the mouse.

*param2* contain the inter-media value of the circular slider.

Inter-media values are not necessarily contiguous.

**CSN\_QUERYBACKGROUNDCOLOR**

This code gives the application the opportunity to set the background color of the circular slider. CLR\_\* or SYSCLR\_\* values can be returned for the background color.

*param2* is NULL.

**param2**

**ulnotifyspec** (ULONG)

Notify control-specific information.

### **Returns**

**ulReserved** (ULONG)

Reserved value.

### **Remarks**

The circular slider control window procedure generates this message and sends it to its owner, informing the owner of this event.

---

## **WM\_CONTROLPOINTER (in Circular Slider Controls)**

For the cause of this message, see WM\_CONTROLPOINTER.

For a description of the parameters, see WM\_CONTROLPOINTER.

### **Remarks**

For the appropriate remarks, see WM\_CONTROLPOINTER.

### **Default Processing**

For the default processing, see WM\_CONTROLPOINTER.

---

## Circular Slider Control Window Messages

This section describes the Circular Slider Control Window Procedure actions on receiving the following messages.

---

### CSM\_QUERYINCREMENT

This message queries the increments used to scroll the value and draw the tick marks.

#### Parameters

**param1**

**ScrollIncr (USHORT)**

The increment value added or subtracted for the value of the control when scrolling.

**param2**

**TickIncr (USHORT)**

The increment value used to draw the tick marks.

#### Returns

**rc (ULONG)**

Success indicator.

TRUE Successful completion

FALSE Errors occurred.

---

### CSM\_QUERYRADIUS

This message queries the current radius of the circular slider.

#### Parameters

**param1**

**uRadius (USHORT)**

The radius of the circular slider.

**param2**

**ulReserved (ULONG)**

Reserved value.

## Returns

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

---

## CSM\_QUERYRANGE

This message queries the value range of the control.

### Parameters

**param1**

**pLow** (PSHORT)

The low range value.

**param2**

**pHigh** (PSHORT)

The high range value.

## Returns

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

---

## CSM\_QUERYVALUE

This message queries the value of the control.

### Parameters

**param1**

**pValue** (PSHORT)

The value of the control.

**param2**

**ulReserved** (ULONG)

Reserved value.

## Returns

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

---

## CSM\_SETBITMAPDATA

This message is used to change the bit maps for the plus and minus buttons. For example, you might want to use left or right arrows. The optimal size for these bit maps is 10 x 10 pels.

## Parameters

**param1**

**pCSBitmapData** (PCSBITMAPDATA)

The structure defining button bit maps.

**param2**

**ulReserved** (ULONG)

Reserved value.

## Returns

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

The optimal size for these bit maps is 10 x 10 pels. Other bit maps are stretched to the necessary size.

---

## CSM\_SETINCREMENT

This message sets the scroll and tick mark increments of the control.

## Parameters

**param1**

**usScrollIncr** (USHORT)

Scroll increment.

This is the number by which the current value is incremented or decremented when one of the circular slider control button is selected.

**param2**

**usTickIncr** (USHORT)

Tick mark increment.

This represents the number of tick marks to "skip" before drawing tick marks around the circular slider.

### **Returns**

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

---

## **CSM\_SETRANGE**

This message sets the range of values which the control sends to the application via CSN\_TRACKING and CSN\_CHANGE messages.

### **Parameters**

**param1**

**Low** (SHORT)

The minimum value of the circular slider.

**param2**

**High** (SHORT)

The maximum value of the circular slider.

### **Returns**

**rc** (ULONG)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

---

## CSM\_SETVALUE

This message sets the current value of the circular slider control.

### Parameters

**param1**

**Value (SHORT)**

The new value to which to set the circular slider.

**param2**

**ulReserved (ULONG)**

Reserved value.

### Returns

**rc (ULONG)**

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

---

## WM\_CHAR (in Circular Slider Controls)

For the cause of this message, see WM\_CHAR.

For a description of the parameters, see WM\_CHAR.

### Remarks

The slider control window procedure responds to this message by sending it to its owner if it has not processed the key stroke. This is the most common means by which the input focus is switched around the various controls in a dialog box.

The keystrokes processed by a circular slider control are:

**Left Arrow** Moves the slider arm left one increment.

**Right Arrow** Moves the slider arm right one increment.

A circular slider control only processes left and right arrow keystrokes. These keys move the slider arm one increment to the left or right.

### Default Processing

For a description of the default processing, see WM\_CHAR.



---

## **WM\_PRESPARAMCHANGED (in Circular Slider Controls)**

For the cause of this message, see WM\_PRESPARAMCHANGED.

### **Parameters**

#### **param1**

##### **attrtype (ULONG)**

Attribute type.

Presentation parameter attribute identity. The following presentation parameters are initialized by the slider control. The initial value of each is shown in the following list:

##### **PP\_FOREGROUND\_COLOR or PP\_FOREGROUND\_COLORINDEX**

Item foreground color; used when displaying text and bit maps. This color is initialized to SYSCLR\_WINDOWTEXT.

##### **PP\_BACKGROUND\_COLOR or PP\_BACKGROUND\_COLORINDEX**

Slider background color; used for entire control as the background. This color is initialized to SYSCLR\_WINDOW.

#### **param2**

##### **ulReserved (ULONG)**

Reserved value, should be 0.

### **Returns**

#### **ulReserved (ULONG)**

Reserved value, must be 0.

### **Remarks**

The application uses this message to notify the slider that a given inherited presentation parameter has changed.

### **Default Processing**

For a description of the default processing, see WM\_PRESPARAMCHANGED.

---

## WM\_QUERYWINDOWPARAMS (in Circular Slider Controls)

For the cause of this message, see WM\_QUERYWINDOWPARAMS.

### Parameters

#### param1

##### **pwndparams (PWNDPARAMS)**

Pointer to a WNDPARAMS window parameter structure.

This structure contains:

##### **status (USHORT)**

Window parameter selection.

Identifies the window parameters that are to be set or queried. Valid values for the slider control are:

**WPM\_CBCTLDATA** Window control data length.

**WPM\_CTLDATA** Window control data.

The flags in the *status* field are cleared as each item is processed. If the call is successful, the *status* field is 0. If any item has not been processed, the flag for that item is still set.

##### **length (USHORT)**

Length of the window text.

##### **text (PSZ)**

Window text.

##### **presparamlength (USHORT)**

Length of presentation parameters.

##### **presparams (PVOID)**

Presentation parameters.

##### **ctldatalength (USHORT)**

Length of window class-specific data.

##### **ctldata (PVOID)**

Window class-specific data.

#### param2

##### **uiReserved (ULONG)**

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

## Remarks

The slider control window procedure responds to this message by returning the information in the buffer provided. If this message is sent to a slider window of another process, the information in, or identified by, the value of the *pwndparams* field must be in memory shared by both processes.

## Default Processing

For a description of the default processing, see WM\_QUERYWINDOWPARAMS.

---

## WM\_SETWINDOWPARAMS (in Circular Slider Controls)

For the cause of this message, see WM\_SETWINDOWPARAMS.

## Parameters

**param1**

**pwndparams** (PWNDPARAMS)

Pointer to a WNDPARAMS window parameter structure.

This structure contains:

**status** (USHORT)

Window parameter selection.

Identifies the window parameters that are to be set or queried. The valid value for the slider control is:

**WPM\_CTLDATA**

Window control data.

The flags in the *status* field are cleared as each item is processed. If the call is successful, the *status* field is 0. If any item has not been processed, the flag for that item is still set.

**length** (USHORT)

Length of the window text.

**text** (PSZ)

Window text.

**presparamslength** (USHORT)

Length of presentation parameters.

**presparams** (PVOID)

Presentation parameters.

*ctldatalength* (USHORT)  
Length of window class-specific data.

*ctldata* (PVOID)  
Window class-specific data.

**param2**

**ulReserved** (ULONG)  
Reserved value, should be 0.

**Returns**

**rc** (BOOL)  
Success indicator.

TRUE     Successful operation  
FALSE    Error occurred.

**Remarks**

If this message is sent to a slider window of another process, the information in, or identified by, the value of the *pwndparams* field must be in memory shared by both processes.

**Default Processing**

For a description of the default processing, see WM\_SETWINDOWPARAMS.



---

## Chapter 26. Value Set Control Window Processing

This system-provided window procedure processes the actions on a value set control (WC\_VALUESET).

---

### Purpose

Like radio buttons, a value set control (WC\_VALUESET window class) is a visual component whose specific purpose is to allow a user to select one choice from a group of mutually exclusive choices. However, unlike radio buttons, a value set can use graphical images (bit maps or icons), as well as colors, text, and numbers, to represent the items that a user can select.

Even though text is supported, a value set's primary purpose is to display choices as graphical images. By using graphical images in a value set, you can preserve space on the display screen. You can also allow the user to see exactly what is being selected instead of having to rely on descriptions of the choices. This allows a user to make a selection faster than if the user had to read a description of each choice. For example, if you want to allow a user to choose from a variety of patterns, you can present those patterns as value set choices instead of having to provide a list of radio buttons with description of each pattern.

If long strings of data are to be displayed as choices, radio buttons should be used. However, for small sets of numeric or textual data information, either a value set or radio buttons can be used.

The value set is designed to be customizable to meet varying application requirements, while providing an easy-to-use user interface component that can be used to develop products that conform to the Common User Access (CUA) user interface guidelines. The application can specify different types of items, sizes, and orientations for its value sets, but the underlying function of the control remains the same. For a complete description of CUA value sets, refer to the *SAA CUA Guide to User Interface Design* and the *SAA CUA Advanced Interface Design Reference*.

---

### Value Set Control Styles

Value set control window styles are set when a value set window is created.

- Set one of the following styles when creating a value set control window. You can override these styles by specifying VIA\_BITMAP, VIA\_ICON, VIA\_TEXT, VIA\_RGB, or VIA\_COLORINDEX attributes for individual value set items.

#### **VS\_BITMAP**

The attribute for each value set item is set to the VIA\_BITMAP value set item attribute, which means the value set treats each item as a bit map unless otherwise specified. This is the default. Figure 26-1 on page 26-2 provides an example of a value set with bit maps.

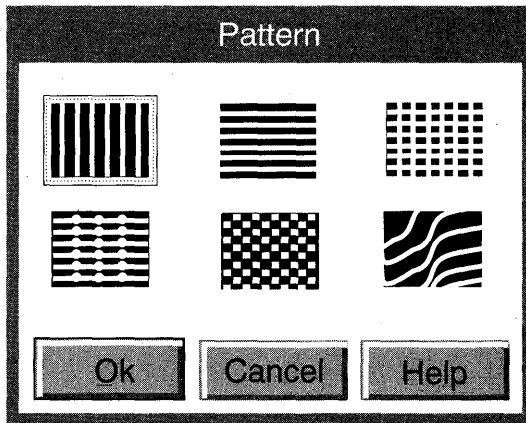


Figure 26-1. Value Set with Bit Maps

### VS\_ICON

The attribute for each value set item is set to the VIA\_ICON value set item attribute, which means the value set treats each item as an icon unless otherwise specified. Figure 26-2 provides an example of a value set with icons.

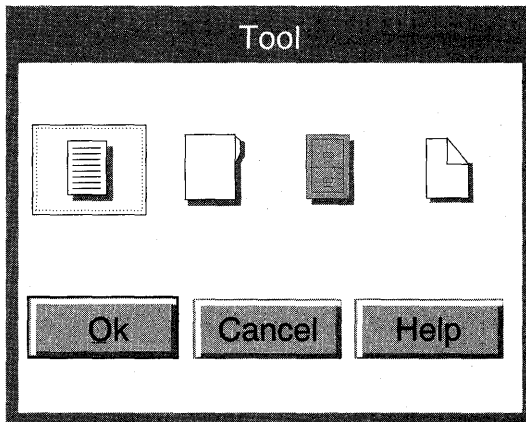


Figure 26-2. Value Set with Icons

### VS\_TEXT

The attribute for each value set item is set to the VIA\_TEXT value set item attribute, which means the value set treats each item as a text string unless otherwise specified. Figure 26-3 on page 26-3 provides an example of a value set with text strings.

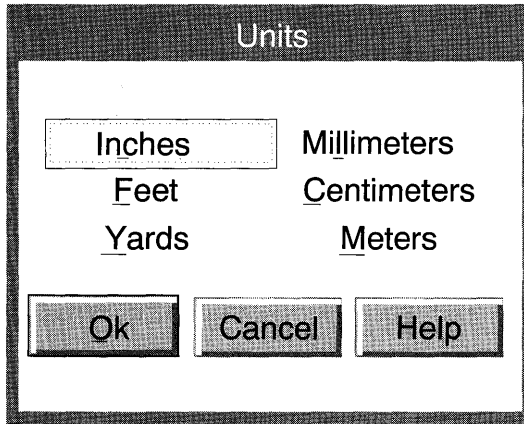


Figure 26-3. Value Set with Text Strings

**VS\_RGB**

The attribute for each value set item is set to the VIA\_RGB value set item attribute, which means the value set treats each item as a RGB color value unless otherwise specified. This style is most often used when you need to create new colors. Figure 26-4 provides an example of a value set with colors.

**VS\_COLORINDEX**

The attribute for each value set item is set to the VIA\_COLORINDEX value set item attribute, which means the value set treats each item as an index into the logical color table unless otherwise specified. This style is most often used when the colors currently available are adequate. Figure 26-4 provides an example of a value set with colors.

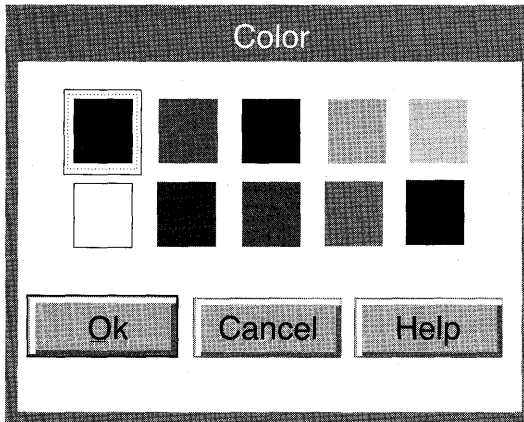


Figure 26-4. Value Set with Colors

- Specify one or more of the following optional window styles, if desired, by using an OR operator (|) to combine them with the style specified from the preceding list:



## VS\_BORDER

The value set draws a thin border around itself to delineate the control. Figure 26-5 on page 26-4 provides an example of a value set with a border.

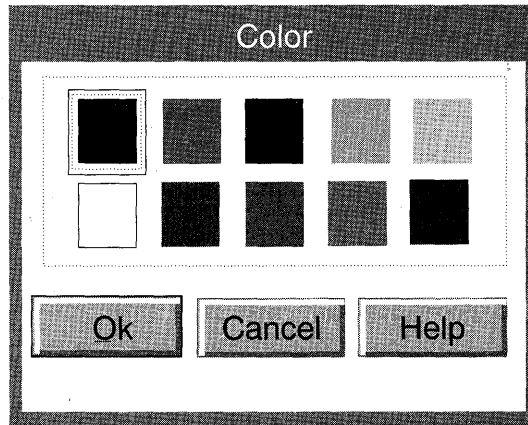


Figure 26-5. Value Set with Border

## VS\_ITEMBORDER

The value set draws a thin border around each item to delineate it from other items.

**Note:** The VS\_ITEMBORDER style is useful for items that are hard to see, such as faint colors or patterns. Figure 26-6 provides an example of a value set with item borders.

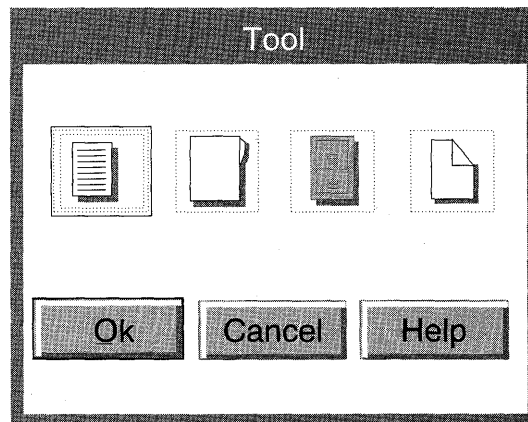


Figure 26-6. Value Set with Item Borders

## VS\_RIGHTTOLEFT

The value set interprets column orientation as right-to-left, instead of the default left-to-right arrangement. This means columns are numbered from right-to-left with the rightmost

column being 1 and counting up as you move left. Home is the rightmost column and end is the leftmost column.

There is no visible difference between a value set ordered left-to-right and a value set ordered right-to-left. Therefore, if your application uses multiple value sets, the ordering of the items should be consistent in each value set to avoid confusing the user.

**Note:** The VS\_RIGHTTOLEFT style is used on creation of the control. Changing this style after creation causes unexpected results.

- VS\_SCALEBITMAPS** The value set automatically scales bit maps to the size of the cell. If this style is not used, each bit map is centered in its cell. Also, if the cell is smaller than the bit map, the bit map is clipped to the size of the cell.
- VS\_OWNERDRAW** The application is notified whenever the background of the value set window is to be painted.

---

## Value Set Control Data

For information on value set control data, see the following:

- "VSCDATA" on page A-204
- "VSDRAGINFO" on page A-205
- "VSDRAGINIT" on page A-205
- "VSTEXT" on page A-206.

---

## Value Set Control Notification Messages

These messages are initiated by the value set control window to notify its owner of significant events.

---

### WM\_CONTROL (in Value Set Controls)

For the cause of this message, see WM\_CONTROL.

#### Parameters

##### param1

##### id (USHORT)

Value set control identity.

##### notifycode (USHORT)

Notify code.

The value set control uses these notification codes:

|              |   |
|--------------|---|
| VN_DRAGLEAVE | The value set receives a DM_DRAGLEAVE message.  |
| VN_DRAGOVER  | The value set receives a DM_DRAGOVER message.   |
| VN_DROP      | The value set receives a DM_DROP message. The VN_DROP notification code is sent only when an item is dropped on an item that has the VIA_DROPONABLE attribute.  |
| VN_DROPHELP  | The value set receives a DM_DROPHELP message.   |
| VN_ENTER     | The user presses the Enter key while the value set window has the focus or double-clicks the select button while the pointer is over an item in the value set.  |
| VN_HELP      | The value set receives a WM_HELP message.   |
| VN_INITDRAG  | The drag button was pressed and the pointer was moved while the pointer was over the value set control. The VN_INITDRAG notification code is sent only for items that have the VIA_DRAGGABLE attribute. |
| VN_KILLFOCUS | The value set is losing the focus.  |
| VN_SELECT    | An item in the value set has been selected and is given selected-state emphasis.  |
| VN_SETFOCUS  | The value set receives the focus.   |

## param2

### **notifyinfo** (ULONG)

Control-specific information.

When the value of the *notifycode* parameter is `VN_DRAGOVER`, `VN_DRAGLEAVE`, `VN_DROP`, or `VN_DROPHELP`, this parameter is a pointer to a `VSDRAGINFO` structure.

When the value of the *notifycode* parameter is `VN_INITDRAG`, this parameter is a pointer to a `VSDRAGINIT` structure.

When the value of the *notifycode* parameter is `VN_ENTER`, `VN_HELP`, or `VN_SELECT`, this parameter contains the row and column of the selection cursor. The low-order word contains the row index, and the high-order word contains the column index.

Otherwise, this parameter is the window handle (HWND) of the value set control.

### **Returns**

#### **ulReserved** (ULONG)

Reserved value, should be 0.

### **Remarks**

The value set control window procedure generates this message and sends it to its owner, informing the owner of this event.

### **Default Processing**

For a description of the default processing, see `WM_CONTROL`.

---

## **WM\_CONTROLPOINTER (in Value Set Controls)**

For the cause of this message, see `WM_CONTROLPOINTER`.

For a description of the parameters, see `WM_CONTROLPOINTER`.

### **Remarks**

For the appropriate remarks, see `WM_CONTROLPOINTER`.

### **Default Processing**

For the default processing, see `WM_CONTROLPOINTER`.

---

## WM\_DRAWITEM (in Value Set Controls)

This notification message is sent to the owner of a value set control each time an item that has the `VIA_OWNERDRAW` attribute is to be drawn, or when the background of a value set window that has the `VS_OWNERDRAW` style bit is to be drawn.

### Parameters

#### param1

**id** (USHORT)

Window identifier.

The window identifier of the value set control sending this notification message.

#### param2

**powneritem** (POWNERITEM)

Pointer to an OWNERITEM data structure.

The following list defines the OWNERITEM data structure fields that apply to the value set control. See OWNERITEM for the default field values.

**hwnd** (HWND)

Value set window handle.

**hps** (HPS)

Presentation-space handle.

**fsState** (ULONG)

Value set window style flags. See "Value Set Control Styles" on page 26-1 for descriptions of these style flags.

**fsAttribute** (ULONG)

Item attribute flags for the indexed item. See "VM\_SETITEMATTR" on page 26-19 for descriptions of these attribute flags.

**fsStateOld** (ULONG)

Reserved.

**fsAttributeOld** (ULONG)

Reserved.

**rclItem** (RECTL)

Item rectangle to be drawn in window coordinates.

**idItem** (LONG)

Identity of component to be drawn.

`VDA_BACKGROUND`

Specifies that a part of the value set background is to be drawn.

`VDA_SURROUNDING`

Specifies that a part of the area surrounding the value set is to be drawn.

VDA\_ITEMBACKGROUND Specifies that the background of an item is to be drawn.

VDA\_ITEM Specifies that an entire item is to be drawn.

*hItem* (ULONG)

If the value of the **identity** parameter is VDA\_ITEMBACKGROUND or VDA\_ITEM, this is the current row and column index of the item to be drawn. The low-order word contains the row index, and the high-order word contains the column index. Otherwise, this is reserved.

## Returns

**rc** (BOOL)

Item-drawn indicator.

TRUE The owner draws the component.

FALSE If the owner does not draw the component, the owner returns this value and the value set control draws the component.

## Remarks

The value set control draws only items that are represented in one of the formats described: text, color, bit maps, or icons.

If an application uses value set controls that contain items that are not represented by the supported formats or requires that the emphasized attribute of an item is to be drawn in a special manner, the application must specify those items as VIA\_OWNERDRAW and those items must be drawn by the owner.

Through this message, the application can provide a custom value set background (the area between the items) and customize the area surrounding the value set (the area on the top and right sides of the value set that is left over when the value set calculates its size). The application can specify how either or both of these areas are drawn and is given the opportunity to do so.

The value set control window procedure generates this message and sends it to its owner, informing the owner that something is to be drawn. The owner is given the opportunity to draw and to indicate whether the value set control should continue with the normal drawing of that component.

## Default Processing

For a description of the default processing, see WM\_DRAWITEM.

---

## Value Set Control Window Messages

This section describes the value set control window procedure actions on receiving the following messages.

---

### VM\_QUERYITEM

This message queries the contents of the item indicated by the values of the *usRow* and *usColumn* fields. The information returned is interpreted based on the attribute of the item.

#### Parameters

##### param1

###### **usRow** (USHORT)

Row index.

Row index of the item to be queried. Rows have a value from 1 to the value of the *usRowCount* field. This value, which is the total number of rows in the value set, is specified in the VSCDATA data structure when the value set control is created.

###### **usColumn** (USHORT)

Column index.

Column index of the item to be queried. Columns have a value from 1 to the value of the *usColumnCount* field. This value, which is the total number of columns in the value set, is specified in the VSCDATA data structure when the value set control is created.

##### param2

###### **pvsText** (PVSTEXT)

Pointer to a VSTEXT data structure or NULL.

If the attribute of the item to query is *VIA\_TEXT*, the value of the *param2* parameter is the same as the value of the *pvsText* field. For all other attributes, the *param2* parameter is reserved and should be set to a NULL value.

See "VSTEXT" on page A-206 for definitions of this structure's fields as they apply to the VM\_QUERYITEM message.

#### Returns

##### **ulItemid** (ULONG)

Item information.

This value depends on the *VIA\_\** attribute specified for the value set item.

- If the *VIA\_TEXT* attribute is set, the following is returned:

###### **usTextLen** (USHORT)

Number of bytes copied to the buffer. This is the length of the text string, excluding the null termination character.

- If the `VIA_BITMAP` attribute is set, the following is returned:  
*hbmlItem* (HBITMAP)  
 Handle of the bit map associated with the item indexed by the *param1* parameter. If the item is empty, a NULL value is returned.
- If the `VIA_ICON` attribute is set, the following is returned:  
*hptItem* (HPOINTER)  
 Handle of the icon associated with the item indexed by the *param1* parameter. If the item is empty, a NULL value is returned.
- If the `VIA_RGB` attribute is set, the following is returned:  
*rgbItem* (ULONG)  
 Color value associated with the item indexed by the *param1* parameter. If the item is empty, a NULL value is returned. Each color value is a 4-byte integer with a value of:  

$$(R * 65536) + (G * 256) + B$$
 where:  
 R    Red intensity value  
 G    Green intensity value  
 B    Blue intensity value.
- If the `VIA_COLORINDEX` attribute is set, the following is returned:  
*ulColorIndex* (ULONG)  
 Index of the color associated with the item indexed by the *param1* parameter.

The following is returned for any of the items to indicate an error condition:

#### VSERR\_INVALID\_PARAMETERS

An error occurred. The `WinGetLastError` function may return the following errors:

- `PMERR_INVALID_PARAMETERS`
- `PMERR_PARAMETER_OUT_OF_RANGE`.

### Remarks

The application uses this message to query the contents of an individual value set item. When querying a text item, the application must provide a buffer for returning the text information. By specifying 0 as the value of the *usBufLen* field and then getting the value returned in the *usTextLen* parameter, an application can determine how large this buffer must be. The value returned is the length of the text string, excluding the null termination character.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.



---

## VM\_QUERYITEMATTR

This message queries the attribute or attributes of the item indicated by the values of the *usRow* and *usColumn* fields.

### Parameters

**param1**

**usRow** (USHORT)

Row index.

Row index of the item for which the attribute or attributes are queried. Rows have a value from 1 to the value of the *usRowCount* field. This value, which is the total number of rows in the value set, is specified in the VSCDATA data structure when the value set control is created.

**usColumn** (USHORT)

Column index.

Column index of the item for which the attribute or attributes are queried. Columns have a value from 1 to the value of the *usColumnCount* field. This value, which is the total number of columns in the value set, is specified in the VSCDATA data structure when the value set control is created.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usItemAttr** (USHORT)

Item information.

This value depends on the *VIA\_\** attribute or attributes specified for the value set item.

- One of the following attributes can be set:

|                |  |
|----------------|--|
| VIA_BITMAP     | If this attribute is set, the item is a bit map. This is the default.        |
| VIA_COLORINDEX | If this attribute is set, the item is an index into the logical color table. |
| VIA_ICON       | If this attribute is set, the item is an icon.                               |
| VIA_RGB        | If this attribute is set, the item is a color entry.                         |
| VIA_TEXT       | If this attribute is set, the item is a text string.                         |

- In addition, one or more of the following attributes can be set:

|                |   |
|----------------|---|
| VIA_DISABLED   | If this attribute is set, the item cannot be selected and is displayed with unavailable-state emphasis, if possible. Unavailable text items are always displayed with unavailable-state emphasis, according to CUA guidelines; for items displayed as color, bit maps, and icons, it is the application's responsibility to determine the best way to show that these items are unavailable, if possible.<br><br>The selection cursor can be moved to an unavailable item by using either the keyboard navigation keys or a pointing device. This allows a user to press the F1 key to find out why that item cannot be selected. |
| VIA_DRAGGABLE  | If this attribute is set, the item can be the source of a direct manipulation action.   |
| VIA_DROPONABLE | If this attribute is set, the item can be the target of a direct manipulation action.   |
| VIA_OWNERDRAW  | If this attribute is set, a paint notification message is sent whenever this item needs painting.   |

- The following is returned if an error occurs:

#### VMERR\_INVALID\_PARAMETERS

The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

### Remarks

The application uses this message to query the specific attribute or attributes of a value set item.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## VM\_QUERYMETRICS

This message queries for the current size of each value set item or for the spacing between items. The value returned is either the width and height of one item, or the spacing between items.

### Parameters

#### param1

##### **fMetric** (USHORT)

Control metric.

Control metric to be queried with this message. This can be either of the following:

**VMA\_ITEMSIZE** If this message attribute is set, the width and height of each item (in pixels) are returned in the *usItemWidth* and *usItemHeight* parameters, respectively.

**VMA\_ITEMSPACING** If this message attribute is set, the horizontal and vertical spacing between items (in pixels) is returned in the *usHorzItemSpacing* parameter and in the *usVertItemSpacing* parameter, respectively.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **ulMetric** (ULONG)

Metric value queried for.

#### **VSERR\_INVALID\_PARAMETERS**

An error occurred. The `WinGetLastError` function may return the following error:

**PMERR\_INVALID\_PARAMETERS.**

**>= 0**

This value depends on the **VMA\_\*** attribute set in the *param1* parameter.

- If the **VMA\_ITEMSIZE** attribute is set, the following is returned:

*usItemWidth* (USHORT)

Width of one value set item, in pixels.

*usItemHeight* (USHORT)

Height of one value set item, in pixels.

- If the `VMA_ITEMSPACING` attribute is set, the following is returned:

*usHorzItemSpacing* (USHORT)

Amount of horizontal space allocated between each value set item, in pixels. This number does not include the space needed for selected-state and target emphasis, and for the selection cursor, because the emphasis and cursor space is automatically allocated by the value set control. The default space amount is 0.

*usVertItemSpacing* (USHORT)

Amount of vertical space allocated between each value set item, in pixels. This number does not include the space needed for selected-state and target emphasis, and for the selection cursor, because the emphasis and cursor space is automatically allocated by the value set control. The default space amount is 0.

### Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

## VM\_QUERYSELECTEDITEM

This message queries for the currently selected value set item indicated by the values of the *usRow* and *usColumn* fields.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ReturnCode**

**usRow** (USHORT)

Row index.

Row index of the currently selected value set item. Rows have a value from 1 to the value of the *usRowCount* field. This value, which is the total number of rows in the value set, is specified in the `VSCDATA` data structure when the value set control is created.

**usColumn (USHORT)**

Column index.

Column index of the currently selected value set item. Columns have a value from 1 to the value of the *usColumnCount* field. This value, which is the total number of columns in the value set, is specified in the VSCDATA data structure when the value set control is created.

**Remarks**

The application uses this message to query the index of the currently selected value set item. If 0 is returned, no item is selected.

**Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return 0.

---

**VM\_SELECTITEM**

This message selects the value set item indicated by the values of the *usRow* and *usColumn* parameters. When a new item is selected, the previously selected item is deselected.

**Parameters**

**param1**

**usRow (USHORT)**

Row index.

Row index of the value set item to select. Rows have a value from 1 to the value of the *usRowCount* field. This value, which is the total number of rows in the value set, is specified in the VSCDATA data structure when the value set control is created.

**usColumn (USHORT)**

Column index.

Column index of the value set item to select. Columns have a value from 1 to the value of the *usColumnCount* field. This value, which is the total number of columns in the value set, is specified in the VSCDATA data structure when the value set control is created.

**param2**

**uiReserved (ULONG)**

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

**TRUE** Item was successfully selected.

**FALSE** An error occurred. The `WinGetLastError` function may return the following errors:

- `PMERR_INVALID_PARAMETERS`
- `PMERR_PARAMETER_OUT_OF_RANGE`.

## Remarks

The application uses this message to select the specified value set item.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return `FALSE`.

---

## VM\_SETITEM

This message specifies the type of information that will be contained by a value set item. This item is indicated by the values of the `usRow` and `usColumn` fields. Each value set item can contain a different type of information. The value set interprets the information set for the item based on the attribute of the item. Value set items that are not set (blank items) are drawn using the background color of the value set.

## Parameters

**param1**

**usRow** (USHORT)

Row index.

Row index of the value set item for which information is being specified. Rows have a value from 1 to the value of the `usRowCount` field. This value, which is the total number of rows in the value set, is specified in the `VSCDATA` data structure when the value set control is created.

**usColumn** (USHORT)

Column index.

Column index of the value set item for which information is being specified. Columns have a value from 1 to the value of the `usColumnCount` field. This value, which is the total number of columns in the value set, is specified in the `VSCDATA` data structure when the value set control is created.

## param2

### **ulltemId** (ULONG)

Item information.

This value depends on the `VIA_*` attribute set for the item.

- If the `VIA_TEXT` attribute is specified, the `ulltemId` field is as follows:

#### *pszItem* (PSZ)

Pointer to a null terminated string containing the text to be placed in the item. If NULL is passed in, the item is blank.

- If the `VIA_BITMAP` attribute is specified, the `ulltemId` field is as follows:

#### *hbmItem* (HBITMAP)

Handle to a bit map that is to be drawn in the item indicated by the `param1` parameter. If NULLHANDLE is passed in, the item will be blank.

- If the `VIA_ICON` attribute is specified, the `ulltemId` field is as follows:

#### *hptItem* (HPOINTER)

Handle to the icon that is to be drawn in the item indicated by the `param1` parameter. If NULLHANDLE is passed in, the item is blank.

- If the `VIA_RGB` attribute is specified, the `ulltemId` field is as follows:

#### *rgbItem* (ULONG)

Color value to be drawn in the item indicated by the `param1` parameter. If an invalid value is passed in (a value greater than 0x00FFFFFF), the item is blank. Each color value is a 4-byte integer with a value of:

$$(R * 65536) + (G * 256) + B$$

where:

- R Red intensity value
- G Green intensity value
- B Blue intensity value.

- If the `VIA_COLORINDEX` attribute is specified, the `ulltemId` field is as follows:

#### *ulColorIndex* (ULONG)

Index of the color in the logical color table to be drawn in the item indicated by the `param1` parameter.

## **Returns**

### **rc** (BOOL)

Success indicator.

**TRUE** Item was successfully set.

**FALSE** An error occurred. The `WinGetLastError` function may return the following errors:

- `PMERR_INVALID_PARAMETERS`
- `PMERR_PARAMETER_OUT_OF_RANGE`.

## Remarks

The application uses this message to set the contents of an individual value set item. To set the values for the entire value set, an application would loop through the rows and columns, setting the value of each item during the initial value set window processing before the window becomes visible.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## VM\_SETITEMATTR

This message sets the attribute or attributes of the item indicated by the values of the *usRow* and *usColumn* parameters.

## Parameters

### param1

#### **usRow** (USHORT)

Row index.

Row index of the value set item for which attributes are being specified. Rows have a value from 1 to the value of the *usRowCount* field. This value, which is the total number of rows in the value set, is specified in the VSCDATA data structure when the value set control is created.

#### **usColumn** (USHORT)

Column index.

Column index of the value set item for which attributes are being specified. Columns have a value from 1 to the value of the *usColumnCount* field. This value, which is the total number of columns in the value set, is specified in the VSCDATA data structure when the value set control is created.

### param2

#### **usItemAttr** (USHORT)

Item attributes.

Attribute or attributes of the item to be set or reset based on the value of the *fSet* field. These attributes can be as follows:

- One of the following attributes can be set:

|                |  |
|----------------|--|
| VIA_BITMAP     | If this attribute is set, the item is a bit map. This is the default.        |
| VIA_COLORINDEX | If this attribute is set, the item is an index into the logical color table. |
| VIA_ICON       | If this attribute is set, the item is an icon.                               |



|  |   |
|--|---|
| VIA_RGB  | If this attribute is set, the item is a color entry.  |
| VIA_TEXT   | If this attribute is set, the item is a text string.  |
| • In addition, one or more of the following attributes can be set: |   |
| VIA_DISABLED   | If this attribute is set, the item cannot be selected and is displayed with unavailable-state emphasis, if possible. Unavailable text items are always displayed with unavailable-state emphasis, according to CUA guidelines; for items displayed as color, bit maps, and icons, it is the application's responsibility to determine the best way to show that these items are unavailable, if possible.<br><br>The selection cursor can be moved to an unavailable item by using either the keyboard navigation keys or a pointing device. This allows a user to press the F1 key to find out why that item cannot be selected. |
| VIA_DRAGGABLE  | If this attribute is set, the item can be the source of a direct manipulation action.   |
| VIA_DROPONABLE   | If this attribute is set, the item can be the target of a direct manipulation action.   |
| VIA_OWNERDRAW  | If this attribute is set, a paint notification message is sent whenever this item needs painting.   |

#### **fSet (USHORT)**

Set or reset flag.

TRUE Set the attribute of the indicated item.

FALSE Turn off the attribute of the indicated item.

#### **Returns**

**rc (BOOL)**

Success indicator.

TRUE Attribute or attributes were set successfully.

FALSE An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

#### **Remarks**

The application uses this message to either set or reset a specific attribute or attributes of a value set item. This provides customization of a control at the item level, so that applications can provide their own types of items with a value set, as well as perform direct manipulation and other actions.

## Default Processing

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

## VM\_SETMETRICS

This message sets the size of each item in the value set control, the spacing between items, or both.

### Parameters

#### param1

##### **fMetric** (USHORT)

Units of measurement.

Unit or units of measurement that are to be set for the value set control. This can be either of the following:

**VMA\_ITEMSIZE** If this message attribute is set, the width and height of each item is set using the values of the *usItemWidth* and *usItemHeight* parameters, respectively.

**VMA\_ITEMSPACING** If this message attribute is set, the horizontal and vertical spacing between each item is set using the values of the *usHorzItemSpacing* and *usVertItemSpacing* parameters, respectively.

#### param2

##### **ulltemId** (ULONG)

Item information.

This value depends on the **VMA\_\*** attribute set for the message.

- If the **VMA\_ITEMSIZE** attribute is specified, the *ulltemId* field is as follows:

##### *usItemWidth* (USHORT)

Width to be set for each value set item, in pixels. The number of pixels specified cannot be less than 2.

##### *usItemHeight* (USHORT)

Height to be set for each value set item, in pixels. The number of pixels specified cannot be less than 2.

- If the **VMA\_ITEMSPACING** attribute is specified, *ulltemId* field is as follows:

##### *usHorzItemSpacing* (USHORT)

Amount of horizontal space to be set between each value set item, in pixels. This number does not include the space needed for selected-state and target emphasis, and for the selection cursor, because the emphasis

and cursor space is automatically set by the value set control. The default spacing is 0.

*usVertItemSpacing* (USHORT)

Amount of vertical space to be set between each value set item, in pixels. This number does not include the space needed for selected-state and target emphasis, and for the selection cursor, because the emphasis and cursor space is automatically set by the value set control. The default spacing is 0.

**Returns**

**rc** (BOOL)

Success indicator.

TRUE Item size or spacing was successfully set.

FALSE An error occurred. The WinGetLastError function may return the following errors:

- PMERR\_INVALID\_PARAMETERS
- PMERR\_PARAMETER\_OUT\_OF\_RANGE.

**Remarks**

Upon receiving this message, the value set redraws the control with the new width, height, and spacing specifications for each item. Any items that do not fit within the current window size are clipped.

When the value set control receives a WM\_SIZE (in Value Set Controls) message, which is sent when the value set window is resized, the value set control defaults the size of each item by dynamically dividing the window size by the number of rows and columns. It allows enough room for the border, selection cursor, and selection emphasis, and defaults the spacing between items to 0. To override these default settings, the application must resend the VM\_SETMETRICS message.

**Default Processing**

The default window procedure does not expect to receive this message and therefore takes no action on it other than to return FALSE.

---

**WM\_CHAR (in Value Set Controls)**

For the cause of this message, see WM\_CHAR.

For a description of the parameters, see WM\_CHAR.

**Remarks**

The value set control window procedure responds to this message by sending it to its owner if it has not processed the key stroke. This is the most common means by which the focus is switched from one control to another in a value set window.

The keystrokes processed by a value set control are:

| <b>Key Name</b>    | <b>Action Performed</b>   |
|--------------------|---|
| <b>Down Arrow</b>  | Moves the selection cursor down one item. When the selection cursor reaches the bottom, the Down Arrow has no effect.   |
| <b>Up Arrow</b>    | Moves the selection cursor up one item. When the selection cursor reaches the top, the Up Arrow has no effect.  |
| <b>Left Arrow</b>  | Moves the selection cursor left one item. When the selection cursor reaches the leftmost column, the Left Arrow has no effect.  |
| <b>Right Arrow</b> | Moves the selection cursor right one item. When the selection cursor reaches the rightmost column, the Right Arrow has no effect.   |
| <b>Home</b>        | Moves the selection cursor to the leftmost column of the value set control (NLS dependent). Pressing the Home key when the leftmost column is selected has no effect. The row index does not change.            |
| <b>End</b>         | Moves the selection cursor to the rightmost column of the value set control (NLS dependent). Pressing the End key when the rightmost column is selected has no effect. The row index does not change.           |
| <b>PgDn</b>        | Moves the selection cursor to the bottom row of the value set control. Pressing the Page Down key when the bottom row is selected has no effect. The column index does not change.                              |
| <b>PgUp</b>        | Moves the selection cursor to the top row of the value set control. Pressing the Page Up key when the top row is selected has no effect. The column index does not change.                                      |
| <b>Ctrl+Home</b>   | Moves the selection cursor to the item in the top row and leftmost column of the value set control (NLS dependent). Pressing the Ctrl+Home keys when the top row and leftmost column is selected has no effect. |
| <b>Ctrl+End</b>    | Moves the selection cursor to the bottom row and rightmost column of the value set control (NLS dependent). Pressing the Ctrl+End keys when the bottom row and rightmost column is selected has no effect.      |
| <b>Enter</b>       | Sends a VN_ENTER notification code to the owner of the value set with the row and column indices of the selected item.  |
| <b>(Mnemonic)</b>  | If the VS_TEXT style bit is set for the value set, any mnemonics specified can be used to select an item.   |

### **Default Processing**

For a description of the default processing, see WM\_CHAR.

---

## WM\_PRESPARAMCHANGED (in Value Set Controls)

For the cause of this message, see WM\_PRESPARAMCHANGED.

### Parameters

#### param1

##### attrtype (ULONG)

Attribute type.

Presentation parameter attribute identity. The following presentation parameters are initialized by the value set control. The initial value of each is shown in the following list:

##### PP\_FOREGROUND\_COLOR or PP\_FOREGROUND\_COLORINDEX

Item foreground color; used when displaying text and bit maps. This color is initialized to SYSCLR\_WINDOWTEXT.

##### PP\_BACKGROUND\_COLOR or PP\_BACKGROUND\_COLORINDEX

Value set background color; used for entire control as the background. This color is initialized to SYSCLR\_WINDOW.

##### PP\_HILITEBACKGROUND\_COLOR or PP\_HILITEBACKGROUND\_COLORINDEX

Selection color; this is the color used for selected-state and target emphasis. This color is initialized to SYSCLR\_HILITEBACKGROUND.

##### PP\_BORDER\_COLOR or PP\_BORDER\_COLORINDEX

Value set and item border color. This color is initialized to SYSCLR\_WINDOWFRAME.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

##### ulReserved (ULONG)

Reserved value, should be 0.

### Remarks

The application uses this message to notify the value set that a given inherited presentation parameter has changed.

### Default Processing

For a description of the default processing, see WM\_PRESPARAMCHANGED.

---

## WM\_QUERYWINDOWPARAMS (in Value Set Controls)

For the cause of this message, see WM\_QUERYWINDOWPARAMS.

### Parameters

#### param1

##### **wndparams** (PWNDPARAMS)

Pointer to a WNDPARAMS window parameter structure.

See WNDPARAMS for descriptions of the default fields. For a value set, the valid values for the *fsStatus* field are WPM\_CBCTLDATA and WPM\_CTLDATA.

The flags in the *fsStatus* field are cleared as each item is processed. If the call is successful, the *fsStatus* field is NULL. If any item has not been processed, the flag for that item is still set.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE     Successful operation.  
FALSE    Error occurred.

### Remarks

The value set control window procedure responds to this message by returning the information in the buffer provided. If this message is sent to a value set window of another process, the information in, or identified by, the *wndparams* parameter must be in memory shared by both processes.

### Default Processing

For a description of the default processing, see WM\_QUERYWINDOWPARAMS.

---

## WM\_SETWINDOWPARAMS (in Value Set Controls)

For the cause of this message, see WM\_SETWINDOWPARAMS.

### Parameters

#### param1

##### **wndparams** (PWNDPARAMS)

Pointer to a WNDPARAMS structure.

See WNDPARAMS for descriptions of the fields. For a value set, the valid value of the *fsStatus* field is WPM\_CTLDATA.

#### param2

##### **ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

#### **rc** (BOOL)

Success indicator.

TRUE Successful operation

FALSE Error occurred.

### Remarks

If this message is sent to a value set window of another process, the information in, or identified by, the *wndparams* parameter must be in memory shared by both processes.

### Default Processing

For a description of the default processing, see WM\_SETWINDOWPARAMS.

---

## WM\_SIZE (in Value Set Controls)

For the cause of this message, see WM\_SIZE.

For a description of the parameters, see WM\_SIZE.

### Remarks

When the value set window is sized, the value set control defaults the size of each item by dynamically dividing the window size by the number of rows and columns. It allows enough room for the border, selection cursor, and selection emphasis, and defaults the spacing between items to 0. To override these default settings, the application must resend the VM\_SETMETRICS message.

## **Default Processing**

For a description of the default processing, see WM\_SIZE.





---

## Chapter 27. Clipboard Messages

### Purpose

The clipboard is used by the end-user to transfer data between Presentation Manager\* (PM\*) applications using the following operations:

- Cut**            Remove from a window, leaving a gap in the source, and save for later use.
- Copy**           Copy from a window, leaving the source intact, and save for later use.
- Paste**           Paste the **cut** or **copied** data into the window of an application (the target).

---

### WM\_DESTROYCLIPBOARD

This message is sent to the clipboard owner when the clipboard is emptied through a call to WinEmptyClipbrd.

#### Parameters

**param1**

**ulReserved (ULONG)**  
Reserved value, should be 0.

**param2**

**ulReserved (ULONG)**  
Reserved value, should be 0.

#### Returns

**ulReserved (ULONG)**  
Reserved value, should be 0.

#### Remarks

If there is any data that has been set with the CFI\_OWNERFREE flag, the clipboard owner must release the data at this time.

#### Default Processing

None.

---

## WM\_DRAWCLIPBOARD

This message is sent to the clipboard viewer window whenever the contents of the clipboard change; that is, as a result of the `WinCloseClipbrd` function following a call to `WinSetClipbrdData`.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Default Processing

None.

---

## WM\_HSCROLLCLIPBOARD

This message is sent to the clipboard-owner window when the clipboard contains a data handle for the `CFI_OWNERDISPLAY` format, and there is an event in the clipboard viewer's horizontal scroll bar.

### Parameters

**param1**

**hwndViewer** (HWND)

Handle.

This contains a handle to the clipboard application window.

**param2**

**sposScroll** (SHORT)

Scroll position.

The position is either:

0        *scroll* is other than `SB_SLIDERPOSITION`

Other The position of the slider when *scodeScroll* is SB\_SLIDERPOSITION.

### **scodeScroll** (SHORT)

Scroll-bar code.

This is one of the SB\_\* scroll-bar codes as defined in WM\_HSCROLL (in Horizontal Scroll Bars).

|                   |   |
|-------------------|---|
| SB_LINELEFT       | Sent if the operator clicks the left arrow of the scroll bar, or presses the VK_LEFT key.                                   |
| SB_LINERIGHT      | Sent if the operator clicks the right arrow of the scroll bar, or presses the VK_RIGHT key.                                 |
| SB_PAGELEFT       | Sent if the operator clicks the area to the left of the slider, or presses the VK_PAGELEFT key.                             |
| SB_PAGERIGHT      | Sent if the operator clicks the area to the right of the slider, or presses the VK_PAGERIGHT key.                           |
| SB_SLIDERPOSITION | Sent to indicate the final position of the slider. <i>sposScroll</i> contains the final position of the slider.             |
| SB_SLIDERTRACK    | Sent every time the slider position changes if the operator moves the scroll bar slider with the pointer device.            |
| SB_ENDSCROLL      | Sent when the operator has finished scrolling, but only if the operator has not been doing any absolute slider positioning. |

### **Returns**

**ulReserved** (ULONG)

Reserved value, should be 0.

### **Remarks**

The clipboard owner is responsible for displaying the clipboard contents. The clipboard owner should use WinInvalidateRect or repaint as desired. The scroll-bar position is also reset.

### **Default Processing**

None.

---

## WM\_PAINTCLIPBOARD

This message is sent when the clipboard contains a data handle with the CFI\_OWNERDISPLAY information flag set.

### Parameters

#### param1

**hwndViewer** (HWND)

Handle.

This is a handle to the clipboard application window.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

As the clipboard owner is responsible for displaying the clipboard contents, this message notifies the clipboard application that its client area needs repainting. The WM\_PAINTCLIPBOARD message is sent to the owner of the clipboard to request repainting of all or part of the client area of the clipboard application.

**Note:** To determine whether the entire client area needs repainting or just a portion of it, the clipboard owner must compare the dimensions of the drawing area to the dimensions given in the most recent WM\_SIZECLIPBOARD message.

### Default Processing

None.

---

## WM\_RENDERALLFMTS

This message is sent to the application that owns the clipboard while the application is being destroyed.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The application renders the clipboard data in all formats it is capable of generating and passes a handle to each format to `WinSetClipboardData`. This ensures that the data in the clipboard can be rendered even though the application has been destroyed.

### Default Processing

None.

---

## WM\_RENDERFMT

This message is a request to the clipboard owner to render the data of the format specified in *usfmt*.

### Parameters

**param1**

**usfmt** (USHORT)

Data format.

This is the format of the data to be rendered.

**CF\_BITMAP**           A bit map.

**CF\_DSPBITMAP**       A bit-map representation of a private data format.

**CF\_DSPMETAFILE**    A metafile representation of a private data format.

**CF\_DSPTTEXT**        A textual representation of a private data format.

**CF\_METAFILE**        A metafile.

**CF\_TEXT**             An array of text characters.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The data is rendered into a global handle, which is then set into the clipboard with `WinSetClipbrdData`.

## Default Processing

None.

---

## WM\_SIZECLIPBOARD

This message is sent when the clipboard contains a data handle for the `CFI_OWNERDISPLAY` format, and the clipboard application window has changed size.

## Parameters

**param1**

**hwndViewer** (HWND)

Handle of viewer window.

**param2**

**ppaint** (PRECTL)

Rectangle to be re-painted.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

The default window procedure takes no action on this message except to set *ulReserved* to 0.

---

## WM\_VSCROLLCLIPBOARD

This message is sent to the clipboard owner window when the clipboard contains a data handle for the CFI\_OWNERDISPLAY format, and there is an event in the clipboard viewer's vertical scroll bar.

### Parameters

#### param1

##### hwndViewer (HWND)

Handle.

This contains a handle to the clipboard application window.

#### param2

##### sposScroll (SHORT)

Scroll position.

The position is either:

0 *scodeScroll* is other than SB\_SLIDERPOSITION

Other The position of the slider when *scodeScroll* is SB\_SLIDERPOSITION.

##### scodeScroll (SHORT)

Scroll-bar code.

This is one of the SB\_\* scroll-bar codes as defined in WM\_HSCROLL (in Horizontal Scroll Bars).

|                   |   |
|-------------------|---|
| SB_LINELEFT       | Sent if the operator clicks the left arrow of the scroll bar, or depresses the VK_LEFT key.                                 |
| SB_LINERIGHT      | Sent if the operator clicks the right arrow of the scroll bar, or depresses the VK_RIGHT key.                               |
| SB_PAGELEFT       | Sent if the operator clicks the area to the left of the slider, or depresses the VK_PAGELEFT key.                           |
| SB_PAGERIGHT      | Sent if the operator clicks the area to the right of the slider, or depresses the VK_PAGERIGHT key.                         |
| SB_SLIDERPOSITION | Sent to indicate the final position of the slider. <i>sposScroll</i> contains the final position of the slider.             |
| SB_SLIDERTRACK    | Sent every time the slider position changes if the operator moves the scroll bar slider with the pointer device.            |
| SB_ENDSCROLL      | Sent when the operator has finished scrolling, but only if the operator has not been doing any absolute slider positioning. |



**Returns****ulReserved** (ULONG)

Reserved value, should be 0.

**Remarks**

The clipboard owner is responsible for displaying the clipboard contents. The clipboard owner should use `WinInvalidateRect` or `repaint` as desired. The scroll bar position is also reset.

**Default Processing**

None.

---

## Chapter 28. Direct Manipulation (Drag) Messages

### Purpose

This section describes the processing that occurs during a direct manipulation operation when the application sends or receives a direct manipulation (DM\_\*) message.

---

### DM\_DISCARDOBJECT

This message is sent to a source that supports the "DRM\_DISCARD" rendering method.

### Parameters

**param1**

**pDragInfo** (PDRAGINFO)

Pointer to the DRAGINFO structure representing the items to be discarded.

**mpparam2**

**ulReserved** (MPARAM)

Reserved value, should be NULL.

### Returns

**ulAction** (ULONG)

Flag.

**DRR\_SOURCE** The source window procedure accepts responsibility for the operation.

**DRR\_TARGET** The target window procedure is to accept responsibility for the operation. The OS/2 shell supports the discarding of dragitems that can be rendered by the DRM\_OS2FILE method.

**DRR\_ABORT** Abort the entire DM\_DROP action.

### Remarks

This message is sent to the source window for the drag action. The source should make a copy of the parameters and return. The source should also create a separate thread to execute the discard action if it responds with DRR\_SOURCE.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action on it, other than to set *ulAction* to the default value of NULL.

---

## DM\_DRAGERROR

This message is sent to the caller of DrgDragFiles or DrgAcceptDroppedFiles when an error occurs during a move or copy operation for a file.

### Parameters

#### param1

##### usError (USHORT)

Error code.

Returned from DosCopy, DosMove, or DosDelete.

##### usOperation (USHORT)

Flag.

Flag indicating the operation that failed.

DFF\_MOVE        DosMove failed.

DFF\_COPY        DosCopy failed.

DFF\_DELETE      DosDelete failed.

#### param2

##### hstr (HSTR)

HSTR of file contributing to the error.

### Returns

#### hstrAction (HSTR)

Action indicator.

DME\_IGNORECONTINUE    Do not retry the operation, but continue with the rest of the files.

DME\_IGNOREABORT        Do not retry the operation, and do not try any other files.

DME\_RETRY                Retry the operation.

DME\_REPLACE             Replace the file at the destination. Used if FALSE is not specified.

Other                    HSTR of new file name to use for retry.

### Remarks

The receiver of this message should return the action that the sender should take.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return FALSE.

---

## DM\_DRAGFILECOMPLETE

This message is sent when a direct manipulation operation on a file or files is complete.

### Parameters

#### param1

**hstr** (HSTR)  
File handle.

#### param2

**usOperation** (USHORT)  
Flags.

- |               |  |
|---------------|--|
| DF_MOVE       | The operation was a move. If this flag is not set, the operation was a copy.                                       |
| DF_SOURCE     | The receiving window was the source of the drag. If this flag is not set, the receiver was the target of the drop. |
| DF_SUCCESSFUL | The drag operation was successful for the file. If this flag is not set, the operation failed.                     |

### Returns

**uiReserved** (ULONG)  
Reserved value, should be 0.

### Remarks

*hstr* is HSTR for the source file if this message is sent by `DrgDragFiles`, and is HSTR for the target file if this message is sent by `DrgAcceptDroppedFiles`.

This message is sent by `DrgDragFiles` to its caller when the move or copy operation is completed, regardless of success or failure. It is also sent by `DrgAcceptDroppedFiles` when a file has been successfully dropped on the caller.

### Default Processing

The `WinDefWindowProc` function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_DRAGLEAVE

This message is sent to a window that is being dragged over when one of these conditions occur:

- The object is dragged outside the boundaries of the window.
- The drag operation is terminated while the object is over the window.

### Parameters

**param1**

**pDraginfo** (PDRAGINFO)

Pointer to the DRAGINFO structure for the drag operation.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message allows for target emphasis and de-emphasis during the direct manipulation process. This message is not sent when a drop occurs. Use DM\_DROP as a signal to remove the target emphasis.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action on it other than to return 0.

---

## DM\_DRAGOVER

This message allows the window under the mouse pointer to determine if the object or objects currently being dragged can be dropped.

*param2* is the pointing device pointer location.

### Parameters

**param1**

**pDraginfo** (PDRAGINFO)

Pointer to the DRAGINFO structure representing the object being dragged.

**param2**

**sxDrop** (SHORT)

X-coordinate of the pointing device pointer in desktop coordinates.

**syDrop** (SHORT)

Y-coordinate of the pointing device pointer in desktop coordinates.

## Returns

**ReturnCode**

**usDrop** (USHORT)

Drop indicator.

DOR\_DROP

Object can be dropped. When this reply is given, *usDefaultOp* must be set to indicate which operation is performed if the user should drop at this location.

DOR\_NODROP

Object cannot be dropped at this time. The target can accept the object in the specified type and format using the specified operation, but the current state of the target will not allow it to be dropped on. The target may change state in the future so that the same object may be acceptable.

DOR\_NODROPOP

Object cannot be dropped at this time. The target can accept the object in the specified type and format, but the current operation is not acceptable. A change in the drag operation may change the acceptability of the object.

DOR\_NEVERDROP

Object cannot be dropped. The target cannot accept the object now and will not change state so that the object will be acceptable in the future. If this response is returned, no more DM\_DRAGOVER messages will be sent to the target until the pointer is moved out of and back into the target window.

**usDefaultOp** (USHORT)

Target-defined default operation.

DO\_COPY    Operation is a copy.

DO\_LINK    Operation is a link.

DO\_MOVE    Operation is a move.

Other       Operation is defined by the application.

This value should be greater than or equal to (>=) DO\_UNKNOWN.

## Remarks

This message is sent to the window that is directly under the hot spot of the mouse pointer during the drag operation when any of the following conditions are met:

- The user moves the mouse.
- A key is pressed.
- A WM\_BUTTON1UP, WM\_BUTTON2UP, WM\_BUTTON3UP, or WM\_ENDDRAG message is received. The message corresponds to *vkTerminate* parameter specified by the call to *DrgDrag* indicating that the drag is ending. In this case the message is sent only if the mouse has moved since the last DM\_DRAGOVER message was sent.

The receiver can gain access to the DRAGINFO structure with *DrgAccessDraginfo*. The acceptability of the dragged objects can be determined by querying the *hstrType* and *hstrRMF* string handles in each of the DRAGITEM structures carried in DRAGINFO structure. In order to accept the drop, the target window must be able to accept *all* of the objects that are being dragged.

The receiver should provide target emphasis for itself. The receiver can use *DrgSetDragPointer* to change the bit map while it is being dragged over. A DM\_DRAGLEAVE or DM\_DROP message will be sent to the target in the future. Target emphasis should be removed at that time.

If *usOperation* in DRAGINFO is DO\_DEFAULT or DO\_UNKNOWN and the target returns DOR\_DROP for *usDrop*, *usDefaultOp* should be set to reflect what the target defines as the default operation. This information is used to provide the appropriate modification to the drag pointer and the target's default operation will be passed in the *usOperation* field of the DRAGINFO structure specified in the DM\_DROP message.

If the value of the *usOperation* field is not DO\_DEFAULT or DO\_UNKNOWN, the *usDefaultOp* parameter is ignored.

**Note:** Lazy drag enabled applications are expected to process this message. It is to be handled in the same manner as the standard drag enabled applications.

## Default Processing

The WinDefWindowProc function returns DOR\_NEVERDROP to the sender of this message.

---

## DM\_DRAGOVERNOTIFY

This message is sent to the source of a drag operation immediately after a DM\_DRAGOVER message is sent to a target window.

*param2* is the target's reply to the DM\_DRAGOVER message.

### Parameters

#### **param1**

##### **pDragInfo** (PDRAGINFO)

Pointer to the DRAGINFO structure that represents the object being dragged.

#### **param2**

Target's reply.

##### **usDrop** (USHORT)

Drop indicator.

##### **usDefaultOp** (USHORT)

Default operation.

Target-defined default operation.

### Returns

#### **ulReserved** (ULONG)

Reserved value.

### Remarks

The source window can use this message to modify its behavior or appearance based on a target window's response to the DM\_DRAGOVER message.

See DM\_DRAGOVER for a description of the target window's possible responses.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and therefore takes no action on it other than to return NULL.



---

## DM\_DROP

This message is sent to the target when the dragged object is dropped.

### Parameters

**param1**

**pDraginfo** (PDRAGINFO)

Pointer to the DRAGINFO structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is sent to the target window directly under the hot spot of the mouse pointer at the completion of a direct-manipulation operation only if DOR\_DROP was returned for the DM\_DRAGOVER message sent to the target window during the drag.

The receiver can obtain access to DRAGINFO structure with DrgAccessDraginfo.

The receiver must immediately remove any target emphasis and post a private message to itself to initiate the data transfer conversations needed to complete the operation.

The receiver can use the *cxOffset* and *cyOffset* fields in the DRAGITEM structure to position the dropped object within its window relative to the drop point. Multiple objects are moved in the same relative position to each other in the target window as they were in the source.

With standard drag, DrgDrag does not return until the drag set is dropped on a target window. Since the source window is the caller of the DrgDrag, it receives the handle of the target window that the drag set is dropped on when DrgDrag returns.

Lazy Drag is slightly different. Since the drag operation is non-modal, the DrgLazyDrag returns as soon as it has completed its initialization of the drag. DM\_DROPNOTIFY is posted to the source window after the drag set is dropped.

When the application receiving the DM\_DROP message has finished all data transfer operations, the target window must free the DRAGINFO structure using DrgFreeDraginfo.

## Default Processing

The WinDefWindowProc function calls DrgDeleteDraginfoStrHandles and DrgFreeDraginfo for *pDraginfo* and returns 0.

---

## DM\_DROPHELP

This message requests help for the current drag operation.

### Parameters

**param1**

**pDraginfo** (PDRAGINFO)

Pointer to the DRAGINFO structure used in the drag operation.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

This message is posted to the target of a drop when F1 is pressed during a direct-manipulation operation, and the drag operation is canceled.

The *usOperation* field of *pDraginfo* can be used to provide help information in the context of the drag operation during which it was requested.

The DM\_DROPHELP message is not supported for lazy drag operations. Since the drag operation is non-modal, the user may request help on anything at any time during the drag. If the application wishes to provide drop help, it must specify the action required to invoke drop help (for example a menu choice), and code the support for it explicitly.

### Default Processing

The WinDefWindowProc function calls DrgDeleteDraginfoStrHandles and DrgFreeDraginfo for *pDraginfo* and returns 0.

---

## DM\_DROPNOTIFY

This message provides the source window with the target window handle and a pointer to the DRAGINFO structure allocated by the source window.

### Parameters

#### param1

##### **pDraginfo** (PDRAGINFO)

Pointer to the DRAGINFO structure allocated by the source window receiving the message.

#### param2

##### **hwndTarget** (HWND)

Handle of the target window that the drag set was dropped on.

**Note:** If *hwndTarget* is equal to zero, the drag is canceled, and the drag set is not dropped. `DrgCancelLazyDrag` posts a `DM_DROPNOTIFY` message with an *hwndTarget* value of zero to the source window.

### Returns

#### returns

##### **ulReserved** (ULONG)

Reserved value, must be 0.

### Remarks

This message is posted to the source window involved in the drag operation when the drag set is dropped on a valid target window.

The source window must examine *hwndTarget* to determine if the target window is the same as the source window. If it is not, the source window must immediately free the DRAGINFO; if the source and target windows are the same, the DRAGINFO must be freed by the target window after completing the post-drop conversation.

**Note:** Lazy drag enabled applications are expected to process this message; standard drag applications are not.

### Default Processing

The default message procedure sets *ulReserved* to 0.

---

## DM\_EMPHASIZETARGET

This message is sent to the caller of `DrgAcceptDroppedFiles` to inform it to either apply or remove target emphasis from itself.

### Parameters

**param1**

**sx** (SHORT)

X-coordinate.

X-coordinate of the pointing device pointer in window coordinates.

**sy** (SHORT)

Y-coordinate.

Y-coordinate of the pointing device pointer in window coordinates.

**usparam2**

**usEmphasis** (USHORT)

Flags.

TRUE    Apply emphasis.

FALSE   Remove emphasis.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Default Processing

The `WinDefWindowProc` function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_ENDCONVERSATION

The target uses this message to notify a source that a drag operation is complete.

### Parameters

**param1**

**ulltemID** (ULONG)

Item ID.

The *ulltemID* from the `DRAGITEM` that was contained within the `DRAGINFO` structure when the object was dropped.

## param2

### ulFlags (ULONG)

Flags.

The flags are set as follows:

|                       |   |
|-----------------------|---|
| DMFL_TARGETSUCCESSFUL | The target successfully completed its portion of the rendering operation. |
| DMFL_TARGETFAIL       | The target failed to complete its portion of the rendering operation.     |

## Returns

### ulReserved (ULONG)

Reserved value, should be 0.

## Remarks

This message is used to inform a source that the target has completed its part of a rendering operation. It is sent by the target to the source.

The target must send this message under any of the following circumstances:

- The target receives a DM\_RENDERCOMPLETE message and will not retry the operation.
- The target completes the rendering operation without involvement from the source.
- The target wants to terminate a rendering operation in progress.
- The target chooses not to render an object that was dropped on it.

## Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_FILERENDERED

This message is sent to the window handling the drag conversation for the caller of DrgDragFiles.

## Parameters

### param1

#### rndf (PRENDERFILE)

Pointer to a RENDERFILE structure.

## param2

### usOperation (USHORT)

Flags.

TRUE     Operation succeeded

FALSE    Operation failed.

## Returns

### ulReserved (ULONG)

Reserved value, should be 0.

## Remarks

This message is sent when the rendering (moving or copying) of a file is complete. The handle of this window is the *hwndDragFiles* field of the RENDERFILE structure sent on DM\_RENDERFILE.

## Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_PRINTOBJECT

This message is sent to a source that supports the "DRM\_PRINT" rendering method when objects are dropped on a printer object.

## Parameters

### param1

#### pDragInfo (PDRAGINFO)

Pointer to the DRAGINFO structure representing the objects to be printed.

### param2

#### pPrintDest (PPRINTDEST)

Pointer to the PRINTDEST structure representing printer object to print to.

The structure contains all the parameters required to call the functions DevPostDeviceModes and DevOpenDC.

## Returns

### **ulAction** (ULONG)

Flag.

- DRR\_SOURCE** The source window procedure/object procedure will take responsibility for the print operation.
- DRR\_TARGET** The target printer object will take responsibility for the print operation (this will only work on objects which are of the pre-registered rendering method; "DRM\_OS2FILE.")
- DRR\_ABORT** Abort the entire DM\_DROP action (do not send any more DM\_PRINTOBJECT messages to any selected source object involved in this DM\_DROP).

## Remarks

This message is sent to the source window procedure. The source window procedure is responsible for interpreting the structure given by *param2*. It should make a copy of all the parameters and then return.

The receiver of this message should create a thread in which to dispatch this message in order to facilitate a prompt reply. The thread can then call DevPostDeviceModes and DevOpenDC as appropriate.

## Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action on it, other than to set *ulAction* to the default value of NULL.

---

## DM\_RENDER

This message is used to request a source to provide a rendering of an object in a specified rendering mechanism and format.

## Parameters

**param1**

**pDxfer** (PDRAGTRANSFER)

Pointer to the DRAGTRANSFER structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

## Remarks

The target sends this message to a source window to request a rendering of an object. If the source returns FALSE, it may set flags in the DRAGTRANSFER structure that tell the target how to perform the rendering operation on its own, or how to retry the operation. If no flags are set, the source will not allow a rendering of the object.

If TRUE is returned, the message was processed by the recipient and the requested rendering will take place. The source will post a DM\_RENDERCOMPLETE message to the target when the rendering is complete.

If FALSE is returned, either the message was not processed by the recipient, or the recipient could not perform the requested rendering. See *fsReply* in DRAGTRANSFER for more information.

## Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_RENDERCOMPLETE

This message is posted by a source to a target window. It informs the target that the source has completed a requested rendering operation.

## Parameters

**param1**

**pDxfer** (PDRAGTRANSFER)

Pointer to the DRAGTRANSFER structure.

**param2**

**usFS** (USHORT)

Flag field.

Flag field indicating successful completion.

DMFL\_RENDERFAIL

The source is unable to perform the rendering operation. The target may be allowed to retry. If the target is allowed to retry and chooses not to, it must send a DM\_ENDCONVERSATION message to the source.



|                  |  |
|------------------|--|
| DMFL_RENDEROK    | The source has completed the rendering operation. When the target completes its part of the rendering operation, it must post a DM_RENDERCOMPLETE message to the source.   |
| DMFL_RENDERRETRY | The source has completed the rendering operation and will allow the target to retry its part of the operation if it fails. This flag can be set in conjunction with either the DMFL_RENDERFAIL or DMFL_RENDEROK flags. |

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

If the rendering operation failed for any reason, the source can allow the target to retry the operation. The source should return to the state it was in when the drop occurred for that object. The target resumes the rendering operation from the beginning.

If the rendering operation encounters a permanent failure, the source should fail the operation and proceed as if the rendering was completed.

If the rendering operation completes successfully, the source should return to the state it was in when the drop occurred for that object. This allows the target to retry the operation if its portion of the rendering failed. The target must post a DM\_ENDCONVERSATION message when either of the following occurs:

- It determines that the rendering operation successfully completed
- It chooses not to retry a rendering operation that failed.

## Default Processing

The WinDefWindowProc function should send a DM\_ENDCONVERSATION message to the window indicated in the *hwndItem* field of the DRAGITEM structure. The message should indicate that the target failed in its part of the rendering operation. Sending the DM\_ENDCONVERSATION message allows the source to release the resources it dedicated to the rendering operation.

---

## DM\_RENDERFILE

This message is sent to the caller of DrgDragFiles to tell it to render a file.

### Parameters

#### param1

**rndf** (PRENDERFILE)

Pointer to a RENDERFILE structure.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Render handling.

TRUE The receiver handled the rendering.

FALSE DrgDragFiles should render this file.

### Remarks

This message is sent when TRUE is specified in DrgDragFiles. The receiver should perform the operation indicated by the TRUE field in the RENDERFILE structure, moving or copying *hstrSource* to *hstrTarget*.

When the operation is complete, a DM\_FILERENDERED message should be sent to *hwndDragFiles* window.

The RENDERFILE structure is allocated temporarily for the receiver of this message. The receiver should make a copy if it needs to use the data in this structure after returning.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return 0.

---

## DM\_RENDERPREPARE

This message tells a source to prepare for the rendering of an object.

### Parameters

**param1**

**pDxfer** (PDRAGTRANSFER)

Pointer to a DRAGTRANSFER structure.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (BOOL)

Success indicator.

**TRUE** The message was processed by the recipient and it is ready to perform the rendering operation. The target of the drop sends a DM\_RENDER message to request the rendering with a specific rendering mechanism and format.

**FALSE** The message either was not processed by the recipient, or it is unprepared to perform the rendering. The *hwndItem* field in DRAGITEM may not be properly initialized, and therefore the target should not send a DM\_ENDCONVERSATION message.

### Remarks

This message must be sent when DC\_PREPARE is on in the DRAGITEM structure.

This message is used to allow the source to create an invisible window to handle the conversation required for the data transfer.

### Default Processing

The WinDefWindowProc function does not expect to receive this message and takes no action other than to return 0.

---

## Chapter 29. Dynamic Data Exchange Messages

### Purpose

This section describes the message part of the DDE protocol, which is a set of guidelines that allows two applications to share data freely between one another; not necessarily driven directly by user input.

**Note:** DDE operates between two specific applications, each of which must be aware of the other, and active.

WinDdeInitiate, WinDdePostMsg, and WinDdeRespond are the functions associated with these messages.

---

### WM\_DDE\_ACK

This message notifies an application of the receipt and processing of a WM\_DDE\_EXECUTE, WM\_DDE\_DATA, WM\_DDE\_ADVISE, WM\_DDE\_UNADVISE or WM\_DDE\_POKE message, and in some cases, of a WM\_DDE\_REQUEST message.

This message is always posted.

### Parameters

#### param1

**hwnd** (HWND)

Window handle of the sender.

#### param2

**pDdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

The acknowledging application modifies the *fsStatus* field to return information about the status of the message received:

|                  |   |
|------------------|---|
| DDE_FACK         | 1=request accepted, 0=request not accepted      |
| DDE_FBUSY        | 1=busy, 0=not busy                              |
| DDE_NOTPROCESSED | Reserved for application-specific return codes  |
| DDE_FAPPSTATUS   | The message was not understood and was ignored. |

An application is expected to set DDE\_FBUSY if it is unable to respond to the request at the time it is received. The DDE\_FBUSY flag is defined only when DDE\_FACK is 0.

*offsItemName* identifies the item for which the acknowledgment is being sent.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

None.

---

## WM\_DDE\_ADVISE

This message (posted by a client application) requests the receiving application to supply an update for a data item whenever it changes.

This message is always posted.

## Parameters

**param1**

**hwnd** (HWND)

Window handle of the sender.

**param2**

**pDdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

Flags in the *fsStatus* field are set as follows:

**DDE\_FACKREQ** If this bit is 1, the receiving (server) application is requested to send its WM\_DDE\_DATA messages with the acknowledgment-requested (DDE\_FACKREQ) bit set. This offers a flow control technique, whereby the client application can avoid overload from incoming WM\_DDE\_DATA messages.

**DDE\_FNODATA** If this bit is 1, the server is requested to send its WM\_DDE\_DATA messages with a zero length data portion. These messages are alarms that tell the client the source data has changed. Upon receiving one of these alarms, the client can choose to call for the latest version of the data by issuing a WM\_DDE\_REQUEST message, or the client can choose to ignore the alarm. This is typically used when there is a significant resource cost associated with actually rendering and/or assimilating the data.

*offszItemName* identifies which data item is being requested.

*usFormat* is the preferred type of data of the client. It must be a registered DDE data format number.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The receiving application is expected to reply with a positive WM\_DDE\_ACK message if it can provide the requested data, or with a negative one if it can not.

## Default Processing

None.

---

## WM\_DDE\_DATA

This message notifies a client application of the availability of data. It is always posted.

## Parameters

### **param1**

#### **hwnd** (HWND)

Window handle of the sender.

### **param2**

#### **pDdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

Flags in the *fsStatus* field are set as follows:

- |               |  |
|---------------|--|
| DDE_FACKREQ   | If this bit is 1, the receiving (client) application is expected to send a WM_DDE_ACK message after the memory object has been processed. If it is 0, the client application should not send a WM_DDE_ACK message. |
| DDE_FRESPONSE | If this bit is 1, this data is offered in response to a WM_DDE_REQUEST message. If it is 0, this data is offered in response to a WM_DDE_ADVISE message.   |

*offszItemName* identifies which data item is available.

*offabData* is the data. The format of the data is a registered DDE data format, identified by the *usFormat* field.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

None.

---

## WM\_DDE\_EXECUTE

This message posts a string to a server application to be processed as a series of commands. The server application is expected to post a WM\_DDE\_ACK message in response.

This message is always posted.

## Parameters

**param1**

**hwnd** (HWND)

Window handle of the server.

**param2**

**pDdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

*offabData* contains the commands to be executed.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Default Processing

None.

---

## WM\_DDE\_INITIATE

This message is sent by an application to one or more other applications, to request initiation of a conversation.

This message is always sent.

## Parameters

### param1

**hwnd** (HWND)

Window handle of the sender.

### param2

**pData** (PDDEINIT)

Pointer to initiation data.

This points to a DDEINIT structure. *pszAppName* is the name of the desired server application; if this is a zero-length string, any application can respond. *pszTopic* is the name of the desired topic; if this is a zero-length string, each responding application responds once for each topic that it can support.

## Returns

**rc** (BOOL)

Success indicator.

TRUE Successful completion

FALSE Error occurred.

## Remarks

Upon receiving this message, all applications with names matching the application name (where specified), that support the topic identified by the topic name, are expected to acknowledge.

A modal window, for example a message box, must not be invoked during the processing of this message.

## Default Processing

The default window procedure frees the segment referenced by *param2*.



---

## WM\_DDE\_INITIATEACK

This message is sent by a server application in response to a WM\_DDE\_INITIATE message, for each topic that the server application wishes to support.

### Parameters

#### param1

**hwnd** (HWND)

Window handle of the sender.

#### param2

**pData** (PDDEINIT)

Pointer to initiation data.

This points to a DDEINIT structure. *pszAppName* is the name of the responding server application; it must not be a zero-length string. *pszTopic* is the name of the topic that the server is willing to support; it must not be a zero-length string.

The DDEINIT structure must be in a shareable segment; it is the responsibility of the receiving window procedure to free this segment.

### Returns

**rc** (BOOL)

Success indicator.

TRUE     Successful completion

FALSE    Error occurred.

### Remarks

A modal window, such as a message box, must not be posted during the processing of this message.

### Default Processing

The default window procedure frees the segment referenced by *param2*.

---

## WM\_DDE\_POKE

This message requests an application to accept an unsolicited data item. It is always posted.

### Parameters

#### param1

**hwnd** (HWND)

Window handle of the sender.

## param2

### **pDdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

*offsItemName* identifies the data item to the receiving application.

*offabData* is the data. The format of the data is a registered DDE data format, identified by the *usFormat* field.

## Returns

### **ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The receiving application is expected to reply with a positive WM\_DDE\_ACK message if it accepts the unsolicited data, or with a negative WM\_DDE\_ACK if it does not.

## Default Processing

None.

---

## WM\_DDE\_REQUEST

This message is posted from client to server, to request that the server provide a data item to the client.

This message is always posted.

## Parameters

### **param1**

#### **hwnd** (HWND)

Window handle of the server.

### **param2**

#### **DdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure. See "DDESTRUCT" on page A-46.

*offsItemName* identifies which data item is being requested.

*usFormat* identifies in which registered DDE data format the data item is to be rendered.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The receiving application is expected to respond with a WM\_DDE\_DATA message, containing the requested data, if possible. Otherwise, it is expected to respond with a negative WM\_DDE\_ACK message.

## Default Processing

None.

---

## WM\_DDE\_TERMINATE

This message is posted by either application participating in a DDE conversation, to terminate that conversation.

This message is always posted.

## Parameters

**param1**

**hwnd** (HWND)

Window handle of the sender.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

Upon receiving this message, an application is expected to post a WM\_DDE\_TERMINATE message in response.

## Default Processing

None.

---

## WM\_DDE\_UNADVISE

This message is posted by a client application to a server application to indicate that the specified item should no longer be updated.

This message is always posted.

### Parameters

#### param1

**hwnd** (HWND)

Window handle of a sender.

#### param2

**DdeStruct** (PDDESTRUCT)

DDE structure.

This points to a dynamic data exchange structure (see "DDESTRUCT" on page A-46). *offsetItemName* identifies which data update request is to be retracted. If this is a zero-length string, data update requests for all items are retracted.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

The receiving application is expected to reply with a positive WM\_DDE\_ACK message if it can honor the request, or a negative one if it cannot.

### Default Processing

None.



---

## Chapter 30. Help Manager Messages

### Purpose

This section describes the processing of messages sent by the Help Manager or applications in response to requests for help by the user.

---

### HM\_ACTIONBAR\_COMMAND

This message is sent to the current active application window by the Help Manager to notify the application when the user selects a tailored action bar item.

#### Parameters

param1

**idCommand** (USHORT)

Identity of the action bar item that was selected.

param2

**uiReserved** (ULONG)

Reserved value, should be 0.

#### Returns

**uiReserved** (ULONG)

Reserved value, should be 0.

#### Default Processing

None.

---

### HM\_CONTROL

This message is sent by the Help Manager to the child of the coverage window to add a control in the control area of a window.

#### Parameters

param1

**usReserve** (USHORT)

Reserved value.

**controlres (USHORT)**

Res number of the control that was selected.

For author-defined push buttons, this is the res identification number that was specified with the push button tag (:pbutton.). For default push buttons, this is the res identification number defined in the PMHELP.H file.

**param2****ulReserved (ULONG)**

Reserved value.

**Returns****ulReserved (ULONG)**

Reserved value, should be 0.

**Remarks**

If an application wants to filter any of the controls, it can subclass the child of the coverage window and intercept this message. If the application does not intercept this message, the Help Manager adds the control to the control area.

**Default Processing**

None.

---

**HM\_CREATE\_HELP\_TABLE**

This message is sent by the application to give the Help Manager a new help table.

**Parameters****param1****pHELPTABLE (PHELPTABLE)**

Help table.

This points to a help table structure; see "HELPTABLE" on page A-108.

**param2****ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

**rc** (ULONG)

Return code.

- 0 The procedure was successfully completed
- Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

## Default Processing

None.

---

## HM\_DISMISS\_WINDOW

This message tells the Help Manager to remove the active help window.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (ULONG)

Return code.

- 0 The help window was successfully removed
- Other There was no associated help window.

See also the values of the *ulErrorCode* parameter of the HM\_ERROR message.

## Remarks

If the user requests help from a primary or secondary window, and then interacts with the primary or secondary window without leaving help, the currently displayed help window might not be appropriate for the application window. This message gives the application the ability to remove that help window.

## Default Processing

None.



---

## HM\_DISPLAY\_HELP

This message tells the Help Manager to display a specific help window.

### Parameters

#### param1

##### **idHelpPanelId** (USHORT)

Identity of the help window.

This points to a USHORT data type.

For a value of the *usTypeFlag* parameter of HM\_PANELNAME.

##### **pszHelpPanelName** (PSZ)

Name of the help window.

This points to a string containing the name of the help window.

#### param2

##### **usTypeFlag** (USHORT)

Flag indicating how to interpret the first parameter.

HM\_RESOURCEID    Indicates the *param1* points to the identity of the help window.

HM\_PANELNAME    Indicates the *param1* points to the name of the help window.

### Returns

#### **rc** (ULONG)

Return code.

0            The window was successfully displayed

Other       See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Remarks

*param1* depends on the value of the *usTypeFlag* parameter.

### Default Processing

None.

---

## HM\_ERROR

This message notifies the application of an error caused by a user interaction.

### Parameters

#### param1

##### **ulErrorCode** (ULONG)

Error code.

A constant describing the type of error that occurred. The application can also receive some of these error constants in the *ulReserved* parameter of messages it has sent to the Help Manager.

The error constants are:

##### **HMERR\_LOAD\_DLL**

The resource DLL was unable to be loaded.

##### **HMERR\_NO\_FRAME\_WND\_IN\_CHAIN**

There is no frame window in the window chain from which to find or set the associated help instance.

##### **HMERR\_INVALID\_ASSOC\_APP\_WND**

The application window handle specified on the *WinAssociateHelpInstance* function is not a valid window handle.

##### **HMERR\_INVALID\_ASSOC\_HELP\_INST**

The help instance handle specified on the *WinAssociateHelpInstance* function is not a valid window handle.

##### **HMERR\_INVALID\_DESTROY\_HELP\_INST**

The window handle specified as the help instance to destroy is not of the help instance class.

##### **HMERR\_NO\_HELP\_INST\_IN\_CHAIN**

The parent or owner chain of the application window specified does not have an associated help instance.

##### **HMERR\_INVALID\_HELP\_INSTANCE\_HDL**

The handle specified to be a help instance does not have the class name of a Help Manager instance.

##### **HMERR\_INVALID\_QUERY\_APP\_WND**

The application window specified on a *WinQueryHelpInstance* function is not a valid window handle.

##### **HMERR\_HELP\_INST\_CALLED\_INVALID**

The handle of the instance specified on a call to the Help Manager does not have the class name of a Help Manager instance.

##### **HMERR\_HELPTABLE\_UNDEFINE**

The application did not provide a help table for context-sensitive help.

**HMERR\_HELP\_INSTANCE\_UNDEFIN**

The help instance handle specified is invalid.

**HMERR\_HELPITEM\_NOT\_FOUND**

Context-sensitive help was requested but the ID of the main help item specified was not found in the help table.

**HMERR\_INVALID\_HELPSUBITEM\_SIZE**

The help subtable item size is less than 2.

**HMERR\_HELPSUBITEM\_NOT\_FOUND**

Context-sensitive help was requested but the ID of the help item specified was not found in the help subtable.

**HMERR\_INDEX\_NOT\_FOUND**

The index is not in the library file.

**HMERR\_CONTENT\_NOT\_FOUND**

The library file does not have any content.

**HMERR\_OPEN\_LIB\_FILE**

The library file cannot be opened.

**HMERR\_READ\_LIB\_FILE**

The library file cannot be read.

**HMERR\_CLOSE\_LIB\_FILE**

The library file cannot be closed.

**HMERR\_INVALID\_LIB\_FILE**

Improper library file provided.

**HMERR\_NO\_MEMORY**

Unable to allocate the requested amount of memory.

**HMERR\_ALLOCATE\_SEGMENT**

Unable to allocate a segment of memory for memory allocation requests from the Help Manager.

**HMERR\_FREE\_MEMORY**

Unable to free allocated memory.

**HMERR\_PANEL\_NOT\_FOUND**

Unable to find the requested help window.

**HMERR\_DATABASE\_NOT\_OPEN**

Unable to read the unopened database.

**param2**

**ulReserved (ULONG)**

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

There is no other way to communicate the error to the application since the user initiated communication, not the application. Other errors which occur when the application sends a message to the Help Manager are returned as the *ulReserved* parameter of the message.

The Help Manager does not display any error messages to the user. Instead, the Help Manager sends or returns all error notifications to the application so that it can display its own messages. This procedure ensures a consistent message interface for all user messages.

## Default Processing

None.

---

## HM\_EXT\_HELP

When the Help Manager receives this message, it displays the extended help window for the active application panel.

## Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (ULONG)

Return code.

0 The extended help window was successfully displayed

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

## Default Processing

None.

---

## HM\_EXT\_HELP\_UNDEFINED

This message is sent to the application by the Help Manager to notify it that an extended help window has not been defined.

### Parameters

param1

**ulReserved** (ULONG)

Reserved value, should be 0.

param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

### Remarks

When the extended help window is requested, the Help Manager searches the help table for its identity. If the extended help window identity associated with the current active window is zero, the Help Manager sends this message to the application to notify it that an extended help window has not been defined. The application then can:

- Ignore the request for help and not display a help window.
- Display its own window.
- Use the HM\_DISPLAY\_HELP message to tell the Help Manager to display a particular window.

### Default Processing

None.

---

## HM\_GENERAL\_HELP

When the Help Manager receives this message, it displays the general help window for the active application window.

### Parameters

param1

**ulReserved** (ULONG)

Reserved value, should be 0.

## param2

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (ULONG)

Return code.

0 The general help window was successfully displayed.

Other See the values of the *ulErrorCode* parameter of the `HM_ERROR` message.

## Default Processing

None.

---

## HM\_GENERAL\_HELP\_UNDEFINED

This message is sent to the application by the Help Manager to notify it that a general help window has not been defined.

## Parameters

### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

### param2

**ulReserved** (ULONG)

Reserved.

0 Reserved value, 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

When the general help window is requested, the Help Manager searches the help table for its identity. If the general help window identity associated with the current active window is zero, the Help Manager sends this message to the application to notify it that a general help window has not been defined. The application can then:

- Ignore the request for help and not display a help window.
- Display its own window.

- Use the HM\_DISPLAY\_HELP message to tell the Help Manager to display a particular window.

### Default Processing

None.

---

## HM\_HELP\_CONTENTS

When the Help Manager receives this message, it displays the help contents window.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (ULONG)

Return code.

0 The help contents window was successfully displayed.

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Default Processing

None.

---

## HM\_HELP\_INDEX

When the Help Manager receives this message, it displays the help index window.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**rc** (ULONG)

Return code.

- 0 The help index window was successfully displayed.
- Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

## Default Processing

None.

---

## HM\_HELPSUBITEM\_NOT\_FOUND

The Help Manager sends this message to the application when the user requests help on a field and it cannot find a related entry in the help subtable.

## Parameters

**param1**

**usContext** (USHORT)

Type of window on which help was requested.

- HLPM\_WINDOW An application window
- HLPM\_FRAME A frame window
- HLPM\_MENU A menu window.

**param2**

**sTopic** (SHORT)

Topic identifier.

For a value of the *usContext* parameter of HLPM\_WINDOW or HLPM\_FRAME:

- window Identity of the window containing the field on which help was requested.
- menu Identity of the submenu containing the field on which help was requested.

**sSubTopic** (SHORT)

Subtopic identifier.

For a value of the *usContext* parameter of HLPM\_WINDOW or HLPM\_FRAME:

- control Control identity of the censored field and on which help was requested.

For a value of the *usContext* parameter of HLPM\_MENU:

- 1 No menu item was selected
- other Menu item identity of the currently selected submenu item on which help was requested.



## Returns

**rc** (BOOL)

Action indicator.

## Remarks

If FALSE is returned from this message, the Help Manager displays the extended help window.

The application has the following options:

- Ignore the notification and not display help for that field or window.
- Display its own window.
- Use the HM\_DISPLAY\_HELP message to tell the Help Manager to display a particular window.

## Default Processing

None.

---

## HM\_INFORM

This message is used by the Help Manager to notify the application when the user selects a hypertext field that was specified with the **reftype=inform** attribute of the **:link.** tag.

## Parameters

**param1**

**idnum** (USHORT)

Window identity.

The identity that is associated with the hypertext field.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

0 Reserved value, zero.

## Default Processing

None.

---

## HM\_INVALIDATE\_DDF\_DATA

The application sends this message to IPF to indicate that the previous DDF data is no longer valid.

### Parameters

#### param1

##### rescount (ULONG)

The count of DDFs to be invalidated.

#### param2

##### resarray (PUSHORT)

Pointer to an array.

The pointer to an array of unsigned 16-bit (USHORT) integers that are the *res* numbers of DDFs to be invalidated.

**Note:** If both param1 and param2 are NULL, then all the DDFs in that page will be invalidated.

### Returns

#### rc (ULONG)

Return code.

0 The procedure was successfully completed.

Other See the values of the *errorcode* parameter of the HM\_ERROR message.

### Remarks

When IPF receives this message, it discards the current DDF data and sends a new HM\_QUERY\_DDF\_DATA message to the object communication window.

This message should be sent to the child of the coveragepage window handle.

### Default Processing

None.

---

## HM\_KEYS\_HELP

This message is sent by the application and informs the help manager to display the keys help window.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (ULONG)

Return code.

- 0 The keys help window was successfully displayed
- Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Remarks

When the Help Manager receives this message, it sends a HM\_QUERY\_KEYS\_HELP message to the active application window. The active application window is the window that was specified when the last HM\_SET\_ACTIVE\_WINDOW message was sent. If no HM\_SET\_ACTIVE\_WINDOW message was issued, then the active application window is the window specified in the WinAssociateHelpInstance call.

The application must return one of the following:

- The identity of a keys help window in the *usHelpPanel* parameter of the HM\_QUERY\_KEYS\_HELP message.
- Zero, if no action is to be taken by the Help Manager for keys help.

### Default Processing

None.

---

## HM\_LOAD\_HELP\_TABLE

The application sends this message to give the Help Manager the module handle that contains the help table, the help subtable, and the identity of the help table.

### Parameters

#### param1

**idHelpTable** (USHORT)

Identity of the help table.

**fsidentityflag** (USHORT)

Help table identity indicator.

0xFFFF Reserved value.

#### param2

**MODULE** (HMODULE)

Resource identity.

Handle of the module that contains the help table and help subtable.

### Returns

**rc** (ULONG)

Return code.

0 The procedure was successfully completed.

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Default Processing

None.

---

## HM\_NOTIFY

This message is used by the application to sub-class and change the behavior or appearance of the help window.

### Parameters

#### param1

**controlres** (USHORT)

Res number of the control that was selected.

For author-defined push buttons, this is the res number that was specified with the push button tag (**:pbutton.**). For default push buttons, this is the res number defined in the PMHELP.H file.

**usReserve (USHORT)**

Reserved value, should be 0.

Reserved for events other than CONTROL\_SELECTED and HELP\_REQUESTED.

**usevent (USHORT)**

The type of event which has occurred.

|                      |   |
|----------------------|---|
| CONTROL_SELECTED     | A control was selected.                       |
| HELP_REQUESTED       | Help was requested.                           |
| OPEN_COVERPAGE       | The coverpage is displayed.                   |
| OPEN_PAGE            | The child window of the coverpage is opened.  |
| SWAP_PAGE            | The child window of the coverpage is swapped. |
| OPEN_INDEX           | The index window is displayed.                |
| OPEN_TOC             | The table of contents window is displayed.    |
| OPEN_HISTORY         | The history window is displayed.              |
| OPEN_LIBRARY         | The new library is opened.                    |
| OPEN_SEARCH_HIT_LIST | The search list displayed.                    |

**param2****ulhwnd (ULONG)**

Window handle of relevant window.

**Returns**

rc (BOOL)

Return code.

TRUE IPF will not format the controls and re-size the window.  
FALSE IPF will process as normal.

**Remarks**

This message is sent to the application to notify it of events that the application would be interested in controlling.

**Default Processing**

None.

---

## HM\_QUERY

This message is sent to IPF by the application to request IPF-specific information, such as the current Instance handle, the active communication object window, the active window, or the group number of the current window.

### Parameters

#### param1

##### **usselectionid** (USHORT)

What is being requested.

This parameter should be specified only if the query is for `HMQW_VIEWPORT` and should otherwise be coded as `NULL`.

Specifies whether a res ID, ID number, or group number is being requested. The value can be any of the following constants:

|                           |  |
|---------------------------|--|
| <code>HMQVP_NUMBER</code> | A pointer to a USHORT that holds the res ID of the window.             |
| <code>HMQVP_NAME</code>   | A pointer to a null-terminated string that holds the ID of the window. |
| <code>HMQVP_GROUP</code>  | The group number of the window.  |

##### **usmessageid** (USHORT)

Type of window queried.

Specifies the type of window to query. The value can be any of the following constants:

|                                 |  |
|---------------------------------|--|
| <code>HMQW_INDEX</code>         | The handle of the index window.  |
| <code>HMQW_TOC</code>           | The handle of the Table of Contents window.  |
| <code>HMQW_SEARCH</code>        | The handle of the Search Hitlist window.   |
| <code>HMQW_VIEWEDPAGES</code>   | The handle of the Viewed Pages window.   |
| <code>HMQW_LIBRARY</code>       | The handle of the Library List window.   |
| <code>HMQW_OBJCOM_WINDOW</code> | The handle of the active communication window.   |
| <code>HMQW_INSTANCE</code>      | The handle of the help instance.   |
| <code>HMQW_COVERPAGE</code>     | The handle of the help manager multiple document interface (MDI) parent window. It is where the secondary windows are contained within the parent window.  |
| <code>HMQW_VIEWPORT</code>      | The handle of the viewport window specified in the low-order word of <code>param1</code> and in <code>param2</code> .<br>When <code>HMQW_VIEWPORT</code> is specified in <code>usmessageid</code> , a value must be specified in |

|                     |   |
|---------------------|---|
|                     | <i>usselectionid</i> to indicate whether a res ID, ID number, or group number is being requested. |
| HMQW_GROUP_VIEWPORT | The group number of the window whose handle is specified in param2.                               |
| HMQW_RES_VIEWPORT   | The res number of the window whose handle is specified in param2.                                 |
| HMQW_ACTIVEVIEWPORT | The handle of the currently active window.  |
| USERDATA            | The previously stored user-data.  |

## param2

### pvoid (PVOID)

Varies, depending on value selected above.

*param2* depends on the value of *param1 usmessageid*.

If *param1 usmessageid* is HMQW\_VIEWPORT, then *param2* is a pointer to the res number, ID, or group ID.

If *param1 usmessageid* is HMQW\_GROUP\_VIEWPORT, then *param2* is the handle of the viewport for which the group number is assigned.

If *param1 usmessageid* is HMQW\_RES\_VIEWPORT, then *param2* is the handle of the viewport for which the res number is requested.

## Returns

### rc (ULONG)

Return code.

0 The procedure was not successfully completed.

Other The handle (HWND), group number (USHORT), or res number (USHORT) of the window, or the user data (USHORT), depending on the value of *param1 usselectionid*.

## Default Processing

None.

---

## HM\_QUERY\_DDF\_DATA

This message is sent to the communication object window by IPF when it encounters the dynamic data formatting (:ddf.) tag.

### Parameters

#### param1

**pageclienthwnd** (HWND)

Client handle.

The client handle of the page that contains the object communication window.

#### param2

**resid** (ULONG)

The res ID associated with the DDF tag.

### Returns

**rc** (HDDF)

Return code.

0 An error has occurred in the application's DDF processing.

Other The DDF handle to be displayed.

**Note:** Once this handle has been returned, the HDDF handle can no longer be used by the application.

### Remarks

Upon receiving this message, the communication object calls DdfInitialize to indicate the start of dynamic data formatting (DDF). Any combination of other DDF calls are then made to describe this data. When this is complete, the communication object finishes processing this message, indicating that the DDF data is complete. After that time, the DDF handle received from DdfInitialize is considered invalid.

### Default Processing

None.



---

## HM\_QUERY\_KEYS\_HELP

When the user requests the keys help function, the Help Manager sends this message to the application.

### Parameters

**param1**

**ulReserved** (ULONG)

Reserved value, should be 0.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**usHelpPanel** (USHORT)

Help panel ID.

The identity of the application-defined keys help window to be displayed.

0 Do nothing

Other Identity of the keys help window to be displayed.

### Remarks

The application responds by returning the identity of the requested keys help window. The Help Manager then displays that help window. Returning 0 in the *usHelpPanel* parameter indicates that the Help Manager should do nothing for the keys help function.

### Default Processing

None.

---

## HM\_REPLACE\_HELP\_FOR\_HELP

This message tells the Help Manager to display the application-defined Help for Help window instead of the Help Manager Help for Help window.

### Parameters

**param1**

**idHelpForHelpPanel** (USHORT)

Identity of the application-defined Help for Help window.

0 Use the Help Manager Help for Help window.

Other Identity of the application-defined Help for Help window.

## **param2**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Remarks**

An application may prefer to provide information that is more specific to itself, rather than the more general help information provided in the Help Manager Help for Help window.

## **Default Processing**

None.

---

## **HM\_REPLACE\_USING\_HELP**

This message tells the Help Manager to display the application-defined Using help window instead of the Help Manager Using help window.

## **Parameters**

### **param1**

#### **idUsingHelpPanel (USHORT)**

The identity of the application-defined Using Help window.

0 Use the Help Manager Using Help window.

Other The identity of the application-defined Using Help window.

### **param2**

#### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Returns**

### **ulReserved (ULONG)**

Reserved value, should be 0.

## **Remarks**

An application may prefer to provide information that is more specific to itself, rather than the more general help information that is provided in the Help Manager Using help window. The guidelines that define the current CUA interface recommend the **Using help** choice be provided in a pull-down menu from the **Help** choice.

## Default Processing

None.

---

## HM\_SET\_ACTIVE\_WINDOW

This message allows the application to change the window with which the Help Manager communicates and the window to which the help window is to be positioned.

### Parameters

#### param1

##### **hwndActiveWindow** (HWND)

The handle of the window to be made active.

Its window procedure receives all messages from the Help Manager until the application changes the active window with another HM\_SET\_ACTIVE\_WINDOW message.

#### param2

##### **hwndRelativeWindow** (HWND)

The handle of the window next to which the help window is to be positioned.

The handle of the application window next to which the Help Manager will position a new help window.

**HWND\_PARENT** This Help Manager defined constant tells the Help Manager to trace the parent chain of the window that had the focus when the user requested help.

**Other** Handle of the window next to which the help window is to be positioned.

### Returns

#### **rc** (ULONG)

Return code.

0 The procedure has been successfully completed.

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Remarks

Normally the Help Manager communicates with the application window with which the Help Manager instance has been associated. The help window is positioned next to this same application window.

If the *hwndActiveWindow* parameter is 0, the *hwndRelativeWindow* parameter is set to 0. That is, if the active window is NULL HANDLE, the relative window is not used.

## Default Processing

None.

---

## HM\_SET\_COVERPAGE\_SIZE

This message is sent to IPF by the application to set the size of the coverage, the window within which all other IPF windows are displayed.

### Parameters

**param1**

**coverpagerectl** (PRECTL)

Pointer to RECTL containing the size of the coverage.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

### Returns

**rc** (ULONG)

Return code.

0        The procedure was successfully completed.

Other    See the values of the *errorcode* parameter of the HM\_ERROR message.

### Remarks

The default size for the coverage of a book is the full width of the screen, while the default size for a help file is one-half the width of the screen.

This message takes effect immediately, changing the size of the coverage. If the coverage is not currently open, the requested size is saved for the next open.

## Default Processing

None.

---

## HM\_SET\_HELP\_LIBRARY\_NAME

This message identifies a list of help window library names to the Help Manager instance.

### Parameters

#### param1

##### pszHelpLibraryName (PSZ)

Library name.

This points to a string that contains a list of help window library names that will be searched by the Help Manager for the requested help window. The names must be separated by a blank.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (ULONG)

Return code.

- 0 The newly specified library successfully replaced the current help window library name.
- Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Remarks

Any subsequent communication to the Help Manager with this message replaces the current list of names with the newly specified list.

When help is requested, the Help Manager will search each library in the list for the requested help window.

### Default Processing

None.

---

## HM\_SET\_HELP\_WINDOW\_TITLE

This message allows the application to change the window text of a help window title.

### Parameters

#### param1

##### pszHelpWindowTitle (PSZ)

Help window title.

This points to a string containing the new Help Window title.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

### Returns

#### rc (ULONG)

Return code.

0 The window title was successfully set.

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

### Default Processing

None.

---

## HM\_SET\_OBJCOM\_WINDOW

This message is sent to IPF by the application to identify the communication object window to which the HM\_INFORM and HM\_QUERY\_DDF\_DATA messages will be sent. This message is not necessary if the communication object does not expect to receive either of these messages.

### Parameters

#### hwndparam1

##### objcomhwnd (HWND)

Handle of the communication object window to be set.

#### param2

##### ulReserved (ULONG)

Reserved value, should be 0.

## Returns

**hwndpreviouswnd** (HWND)

The handle of the previous communication object window.

## Remarks

HM\_INFORM and HM\_QUERY\_DDF\_DATA messages which are not processed must be passed to the previous communication object window which was returned when HM\_SET\_OBJECT\_WINDOW was sent.

## Default Processing

None.

---

## HM\_SET\_SHOW\_PANEL\_ID

This message tells the Help Manager to display, hide, or toggle the window identity for each help window displayed.

## Parameters

**param1**

**fsShowPanelId** (USHORT)

The show window identity indicator.

CMIC\_HIDE\_PANEL\_ID

Sets the show option off and the window identity is not displayed.

CMIC\_SHOW\_PANEL\_ID

Sets the show option on and the window identity is displayed.

CMIC\_TOGGLE\_PANEL\_ID

Toggles the display of the window identity.

**param2**

**ulReserved** (ULONG)

Reserved value, should be 0.

**rc** (ULONG)

Return code.

0 The show window identity indicator was successfully changed.

Other See the values of the *ulErrorCode* parameter of the HM\_ERROR message.

## Default Processing

None.

---

## HM\_SET\_USERDATA

The application sends this message to IPF to store data in the IPF data area.

### Parameters

#### param1

**ulReserved** (ULONG)

Reserved value, should be 0.

#### param2

**usrdata** (VOID)

4-byte user data area.

**rc** (ULONG)

Return code.

TRUE The user data was successfully stored.

FALSE The call failed.

### Default Processing

None.

---

## HM\_TUTORIAL

The Help Manager sends this message to the application window when the user selects the Tutorial choice from a help window.

### Parameters

#### param1

**pszTutorialName** (PSZ)

Default tutorial name.

This points to a string that contains the name of the default tutorial program specified in the Help Manager initialization structure. A tutorial name specified in the help window definition overrides this default tutorial program.

#### param2

**ulReserved** (ULONG)

Reserved value, should be 0.



## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The application then calls its own tutorial program.

## Default Processing

None.

---

## HM\_UPDATE\_OBJCOM\_WINDOW\_CHAIN

This message is sent to the currently active communication object by the communication object who wants to withdraw from the communication chain.

## Parameters

**param1**

**hwnd** (HWND)

The handle of the object to be withdrawn from the communication chain.

**param2**

**hwnd** (HWND)

Window containing the handle of the object to be replaced.

## Returns

**ulReserved** (ULONG)

Reserved value, should be 0.

## Remarks

The object that receives this message should check to see if the object handle returned from HM\_SET\_OBJCOM\_WINDOW is equal to the handle in *param1*. If the handle is equal, then the handle in *param1* should be replaced by the handle in *param2*. If the handle is not equal *and* the handle previously received is not NULL HANDLE, then send HM\_UPDATE\_OBJCOM\_WINDOW\_CHAIN to that object.

## Default Processing

None.

---

## Chapter 31. Resource Files

This chapter describes the syntax for the resource language using railroad syntax, and describes the formats used.

Resource files are used to build dialog templates, menu templates, accelerator tables, extended attribute association tables, keyboard scancode mapping tables, keyboard names and fonts. The files must be compiled before they can be used by application programs.

---

### How to Read the Syntax Definitions

Throughout this book, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The  $\blacktriangleright$ — symbol indicates the beginning of a statement.

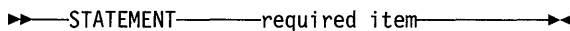
The — $\blacktriangleright$  symbol indicates that the statement syntax is continued on the next line.

The  $\blacktriangleright$ — symbol indicates that a statement is continued from the previous line.

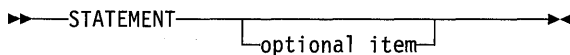
The — $\blacktriangleleft$  symbol indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the  $\blacktriangleright$ — symbol and end with the — $\blacktriangleright$  symbol.

- Required items appear on the horizontal line (the main path).

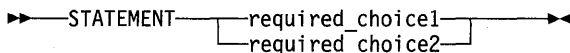


- Optional items appear below the main path.

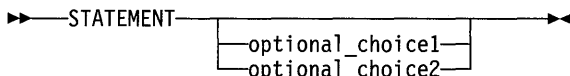


- If a choice can be made from two or more items, they appear vertically, in a stack.

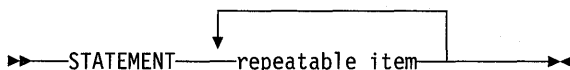
If one of the items *must* be chosen, one item of the stack appears on the main path.



If choosing one of the items is optional, the entire stack appears below the main path.



- An arrow returning to the left above the main path indicates an item that can be repeated.



A repeat arrow above a stack indicates that a choice can be made from the stacked items, or a single choice can be repeated.

- Keywords appear in uppercase, (for example, PARM1). They must be spelled exactly as shown. Variables appear in all lowercase letters (for example: parmX). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, they must be entered as part of the syntax.

---

## Definitions Used in all Resources

The definitions used in all resources are defined in Specification of Values and Resource Load and Memory Options.

### Specification of Values

These rules apply to values specified in resources:

- Coordinates must be integers. There must be no space between the sign of the value and the value itself. For example, “-1” is allowed but “- 1” is not.
- Resource identifiers must be positive integers or names that resolve to positive integers.
- Real values, containing a decimal point, cannot be used.

### Resource Load and Memory Options

The following options define when each resource is loaded and how memory is allocated for each resource.

|                    |  |
|--------------------|--|
| <b>LOADOPTION</b>  | Resource loading options.                      |
| <b>PRELOAD</b>     | Resource is loaded immediately.                |
| <b>LOADONCALL</b>  | Resource is loaded when called.                |
| <b>MEMOPTION</b>   | Resource memory options.                       |
| <b>FIXED</b>       | Resource remains at a fixed memory location.   |
| <b>MOVEABLE</b>    | Resource can be moved if necessary to compact. |
| <b>DISCARDABLE</b> | Resource can be discarded if no longer needed. |
| <b>SEGALIGN</b>    | Resources are aligned on 64K byte boundaries.  |

---

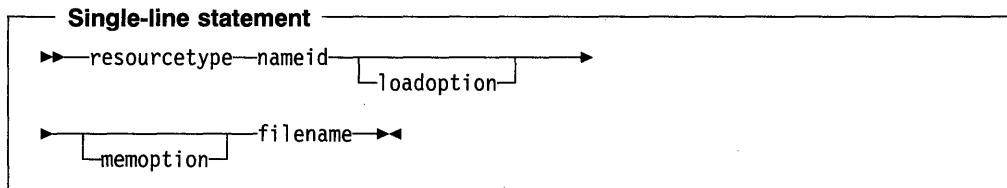
### Resource Script File Specification

The resource script file defines the names and attributes of the resources to be added to the executable file of the application. The file consists of one or more resource statements that define the resource type and original file, if any. See the following for a description of the resource statements:

- Single-Line Statements
- User-Defined Resources
- Directives
- Multiple-Line Statements.

## Single-Line Statements

The general form for all single-line statements is:



### resourcetype (*USHORT*)

One of the following keywords, specifying the type of resource to be loaded:

| Keyword            | Resource type  |
|--------------------|--|
| <b>BITMAP</b>      | A bit-map resource is a custom bit map that an application intends to use in its screen display or as an item in a menu.                               |
| <b>DEFAULTICON</b> | This keyword installs the filename.ico icon definition under the ICON EA of the program file.<br>Example:<br><br>DEFAULTICON <filename.ico>            |
| <b>DLGINCLUDE</b>  | This statement tells the dialog editor which file to use as an include file for the dialogs in the resource file. The <b>nameid</b> is not applicable. |
| <b>FONT</b>        | A font resource is a file containing a font.   |
| <b>ICON</b>        | An icon resource is a bit map defining the shape of the icon to be used for a given application.   |
| <b>POINTER</b>     | A pointer resource is a bit map defining the shape of the pointing device pointer on the display screen.   |

### nameid (*USHORT*)

is either a unique name or an integer number identifying the resource. For a FONT resource, the **nameid** must be a number; it cannot be a name.

### loadoption (*LOADOPTION*)

The default is LOADONCALL.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

### memoption (*MEMOPTION*)

The default is MOVEABLE and DISCARDABLE for POINTER, ICON, and FONT resources. The default for BITMAP resources is MOVEABLE. The FIXED option overrides both MOVEABLE and DISCARDABLE. The SEGALIGN option can be specified independently of other options, if it is not present the default (for all resources) is that the resource is not aligned on a 64KB boundary.

See "Resource Load and Memory Options" on page 31-2 for a description of *MEMOPTION*.

**filename (PCH)**

An ASCII string specifying the OS/2\* name of the file containing the resource. A full path must be given if the file is not in the current working directory.

**Example**

```
POINTER pointer point.cur  
POINTER pointer DISCARDABLE point.cur  
POINTER 10 custom.cur
```

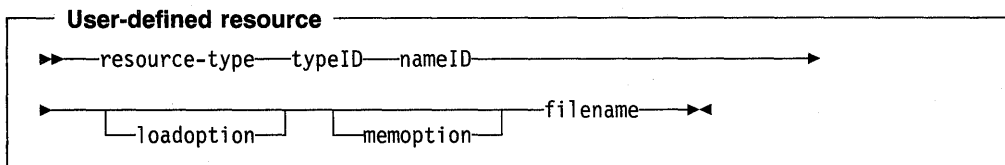
```
ICON desk desk.ico  
ICON desk DISCARDABLE desk.ico  
ICON 11 custom.ico
```

```
BITMAP disk disk.bmp  
BITMAP disk DISCARDABLE disk.bmp  
BITMAP 12 custom.bmp
```

```
FONT 5 CMROMAN.FNT
```

## User-Defined Resources

An application can also define its own resource. The resource can be any data that the application intends to use. A user-defined resource statement has the form:



**typeID**

Either a unique name or an integer number identifying the resource type. If a number is given, it must be greater than 255. The type numbers 1 through 255 are reserved for existing and future predefined resource types. Value 1000 is reserved for custom fonts.

**nameID**

Either a unique name or an integer number identifying the resource.

**loadoption (LOADOPTION)**

The default is LOADONCALL.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

**memoption (MEMOPTION)**

The default is MOVEABLE.

See "Resource Load and Memory Options" on page 31-2 for a description of *MEMOPTION*.

**filename**

An ASCII string specifying the OS/2\* name of the file containing the cursor bit map. A full path must be given if the file is not in the current working directory.

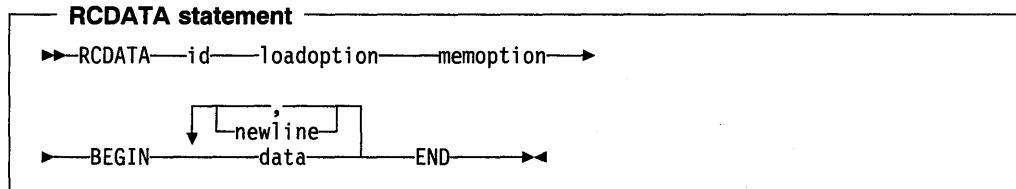
When the resource compiler (RC.EXE) encounters one or more font resources, or any custom resource having type-id of 1000, it creates a font directory resource which it adds to the output binary data.

**Example**

```
RESOURCE MYRES array DATA.RES
RESOURCE 300 14 CUSTOM.RES
```

**RCDATA statement**

The RCDATA statement is provided to allow an application to define a simple data resource.



**id** Either a unique name or an integer number identifying the resource.

**loadoption (LOADOPTION)**

The default is `LOADONCALL`.

See "Resource Load and Memory Options" on page 31-2 for a description of `LOADOPTION`.

**memoption (MEMOPTION)**

The default is `MOVEABLE`.

See "Resource Load and Memory Options" on page 31-2 for a description of `MEMOPTION`.

**data**

A number or string.

## Example:

```
RCDATA 4
BEGIN
  "Sample string."
  "TEST DATA."
  "A message."
END
```

## Directives

The resource directives are special statements that define actions to perform on the file before it is compiled. The directives can assign values to names, include the contents of files, and control compilation of the file.

### #include filename

### rcinclude filename

These directives copy the contents of the file specified by **filename** into the resource before it is compiled. If **rcinclude** is used, the entire file is copied. If **#include** is used, only **#define** statements are copied.

**Note:** If an **rcinclude** is to be commented out, the open comment (**/\***) must appear on the same line as the directive.

**Filename** is an ASCII string. A full path must be given if the file is not in the current directory or in the directory specified by the INCLUDE environment variable. The file extensions .I and .TMP must not be used as these are reserved for system use.

The **filename** parameter is handled as a C string, and two back-slashes must be given wherever one is expected in the path name (for example, root\\sub.) or, a single forward slash (/) can be used instead of double back-slashes (for example, root/sub.)

### Example:

```
#include "wincalls.h"

MENU PenSelect
BEGIN
  MENUITEM "black pen", BLACK_PEN
END
```

Files included in resource script files constants that use **#define** statements may not include any casting of those constants that are used in the resource script. The resource compiler does not parse this casting syntax. For example, the following statement may not be included:

```
#define IDBUTTON1 (USHORT) 3
```

If casting is required for C source compilation, you may use two statements such as:

```
#define IDBUTTON1 3
#define CSRC_IDBUTTON1 ((USHORT)IDBUTTON1)
```

### **#define name value**

This directive assigns the given value to **name**. All subsequent occurrences of **name** are replaced by the value.

**name** is any combination of letters, digits, or punctuation.

**value** is any integer, character string, or line of text.

#### **Example:**

```
#define nonzero 1
#define USERCLASS "MyControlClass"
```

### **#undef name**

This directive removes the current definition of **name**. All subsequent occurrences of **name** are processed without replacement.

**name** is any combination of letters, digits, or punctuation.

#### **Example:**

```
#undef nonzero
#undef USERCLASS
```

### **#ifdef name**

This directive performs a conditional compilation of the resource file by checking the specified name. If the name has been defined using a `#define` directive, `#ifdef` directs the resource compiler to continue with the statement immediately after it. If the name has not been defined, `#ifdef` directs the compiler to skip all statements up to the next `#endif` directive.

**name** is the name to be checked by the directive.

#### **Example:**

```
#ifdef Debug
FONT 4 errfont.fnt
#endif
```

### **#ifndef name**

This directive performs a conditional compilation of the resource file by checking the specified name. If the name has not been defined or if its definition has been removed using the `#undef` directive, `#ifndef` directs the resource compiler to continue processing



statements up to the next `#endif`, `#else`, or `#elif` directive, then skip to the statement after the `#endif`. If the name is defined, `#ifndef` directs the compiler to skip to the next `#endif`, `#else`, or `#elif` directive.

**name** is the name to be checked by the directive.

**Example:**

```
#ifndef Optimize
FONT 4 errfont.fnt
#endif
```

**#if constant expression**

This directive performs a conditional compilation of the resource file by checking the specified constant-expression. If the constant-expression is nonzero, `#if` directs the resource compiler to continue processing statements up to the next `#endif`, `#else`, or `#elif` directive, then skip to the statement after the `#endif`. If the constant-expression is zero, `#if` directs the compiler to skip to the next `#endif`, `#else`, or `#elif` directive.

**constant expression** is a defined name, an integer constant, or an expression consisting of names, integers, and arithmetic and relational operators.

**Example:**

```
#if Version<3
FONT 4 errfont.fnt
#endif
```

**#elif constant expression**

This directive marks an optional clause of a conditional compilation block defined by an `#ifdef`, `#ifndef`, or `#if` directive. The directive carries out conditional compilation of the resource file by checking the specified constant-expression. If the constant-expression is nonzero, `#elif` directs the resource compiler to continue processing statements up to the next `#endif`, `#else`, or `#elif` directive, then skip to the statement after the `#endif`. If the constant-expression is zero, `#elif` directs the compiler to skip to the next `#endif`, `#else`, or `#elif` directive. Any number of `#elif` directives can be used in a conditional block.

**constant expression** is a defined name, an integer constant, or an expression consisting of names, integers, and arithmetic and relational operators.

**Example:**

```
#if Version<3
FONT 4 italic.fnt
#elif Version<7
FONT 4 bold.fnt
#endif
```

## **#else**

This directive marks an optional clause of a conditional compilation block defined by an `#ifdef`, `#ifndef`, or `#if` directive. The `#else` directive must be the last directive before `#endif`.

### **Example:**

```
#ifdef Debug
FONT 4 italic.fnt
#else
FONT 4 bold.fnt
#endif
```

## **#endif**

This directive marks the end of a conditional compilation block defined by an `#ifdef`, `#ifndef`, or `#if` directive. One `#endif` is required for each `#ifdef`, `#ifndef`, and `#if` directive.

## **Multiple-Line Statements**

This sections covers “Code Page Flagging,” “Keyboard Resources” on page 31-10, and the following multiple-line statements:

- “ACCELTABLE Statement” on page 31-10
- “ASSOCTABLE Statement” on page 31-12
- “MENU Statement” on page 31-16
- “STRINGTABLE Statement” on page 31-22
- “Dialog and Window Template Statements” on page 31-13

**Code Page Flagging:** The `CODEPAGE` statement may be placed within the source, to set the code page used for these resources:

- ACCELTABLE
- MENU
- STRINGTABLE
- DIALOGTEMPLATE and WINDOWTEMPLATE.

The `CODEPAGE` statement cannot be encoded within any other statement. All items following a `CODEPAGE` statement are assumed to be in that code page. The code page is encoded in the resource, and the data in the resource is assumed to be in the specified code page. However, no checking is performed.

These code pages can be specified:

- 437
- 850
- 860
- 863
- 865.

If the code page is not specified, code page 850 is assumed.

## Keyboard Resources

**RT\_FKALONG** (=17), is defined in BSEDOS.H, and the resource compiler (RC.EXE) recognizes **FKALONG**. This type identifies a 256-byte table, that can be used for either primary or secondary scan-code mapping.

The resource ID contains three bytes, the least significant byte identifying the type of scan-code mapping table as follows:

- 0 Primary scan-code mapping
- 1 Secondary scan-code mapping.

The other two bytes are 0 for the primary mapping table, and the keyboard ID (as defined in PMWINP.H) for secondary mapping tables. This is to enable simple support to be provided for future keyboards with conflicting scan codes.

The primary scan-code mapping table in the interrupt handler is stored as a resource of this type. The secondary scan-code mapping table in the interrupt handler is also stored as a resource of this type.

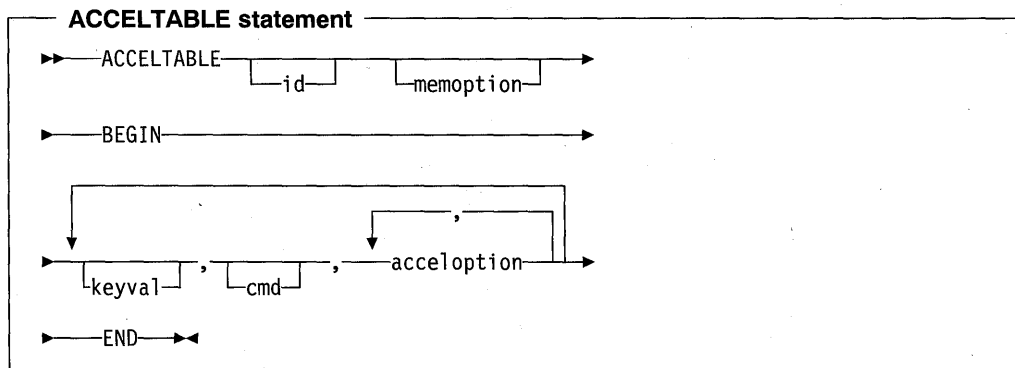
Depending on which keyboard is attached, the resources are loaded when the system is initialized, and transferred to RING-0 byte arrays, where they can be accessed by the interrupt handler as necessary. A default primary scan-code mapping table is transferred if the resource cannot be loaded.

### ACCELTABLE Statement

The ACCELTABLE statement defines a table of *accelerator* keys for an application.

An accelerator is a keystroke defined by the application to give the user a quick way to perform a task. The WinGetMsg function automatically translates accelerator messages from the application queue into WM\_COMMAND, WM\_HELP, or WM\_SYSCOMMAND messages.

The ACCELTABLE statement has the form:



**id** (*USHORT*)

The resource identifier.

## **memoption**

Optional. It consists of the following keyword or keywords, specifying whether the resource is fixed or movable, and whether it can be discarded:

FIXED                   Resource remains at a fixed memory location.  
MOVEABLE               Resource can be moved if necessary to compact memory.  
DISCARDABLE            Resource can be discarded if no longer needed.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

## **keyval (USHORT)**

The accelerator character code. This can be either a constant or a quoted character. If it is a quoted character, the CHAR acceloption is assumed. If the quoted character is preceded with a caret character (^), a control character is specified as if the CONTROL **acceloption** had been used.

## **cmd (USHORT)**

The value of the WM\_COMMAND, WM\_HELP, or WM\_SYSCOMMAND message generated from the accelerator for the indicated key.

## **acceloption (BIT\_16)**

Defines the kind of accelerator.

The following options are available:

ALT  
CHAR  
CONTROL  
HELP  
LONEKEY  
SCANCODE  
SHIFT  
SYSCOMMAND  
VIRTUALKEY.

The VIRTUALKEY, SCANCODE, LONEKEY, and CHAR acceloptions specify the type of message that matches the accelerator. Only one of these options can be specified for each accelerator. For information on the corresponding KC\_\* values, see WM\_CHAR.

The **acceloptions** SHIFT, CONTROL, and ALT, cause a match of the accelerator only if the corresponding key is down.

If there are two accelerators that use the same key with different SHIFT, CONTROL, or ALT options, the more restrictive accelerator should be specified first in the table. For example, Shift-Enter should be placed before Enter.

The SYSCOMMAND **acceloption** causes the keystroke to be passed to the application as a WM\_SYSCOMMAND message. The HELP **acceloption** causes the keystroke to be passed to the application as a WM\_HELP message. If neither is specified, a WM\_COMMAND message is used.

## **Example:**

```

ACCELTABLE MainAcc
BEGIN
    VK_F1,101,HELP
    VK_F3,102,SYSCOMMAND
END

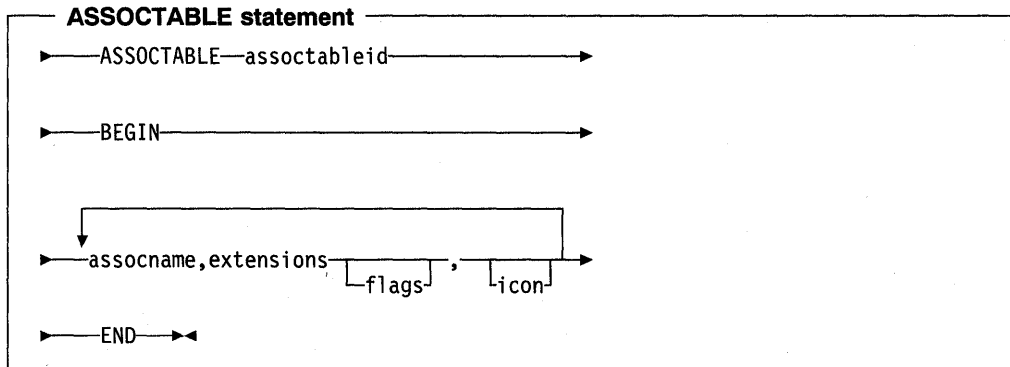
```

This generates a WM\_HELP with value 101 from VIRTUALKEY accelerator F1 and a WM\_SYSCOMMAND with value 102 from VIRTUALKEY accelerator F3.

## ASSOCTABLE Statement

The ASSOCTABLE statement defines the extended attributes (EA) for an application.

The ASSOCTABLE statement has the form:



The source for the ASSOCTABLE description is contained in the resource file for a particular project:

```

ASSOCTABLE assoctableid
BEGIN
"association name", "extension", flags, icon filename
"association name", "extension", flags, icon filename
...
END

```

### association name

Program recognizes data files of this EA TYPE. This is the same name found in the TYPE field of data files.

### assoctableid

A name or number used to identify the assoctable resource.

### extension

3 letter file extension that is used to identify files of this type if they have no EA TYPE entry. (This may be empty.)

### flags

### **EAF\_DEFAULTOWNER**

The default application for the file.

### **EAF\_UNCHANGEABLE**

This flag is set if the entry in the ASSOCTABLE is not to be edited.

### **EAF\_REUSEICON**

This flag is specified if a previously defined icon in the ASSOCTABLE is to be reused. Entries with this flag set have no icon data defined. The icon used for this entry is the icon used for the previous entry (see below). Note that EAF\_\* flags may be ORed together when specified in the ASSOCTABLE.

### **icon filename**

Filename of the icon used to represent this file type. (This may be empty.)

### **Example**

```

ASSOCTABLE 3000
BEGIN
"Product XYZ Spreadsheet", "xys", EAF_DEFAULTOWNER, xyzspr.ico
"Product XYZ Chart", "xyc", EAF_DEFAULTOWNER | EAF_REUSEICON
END

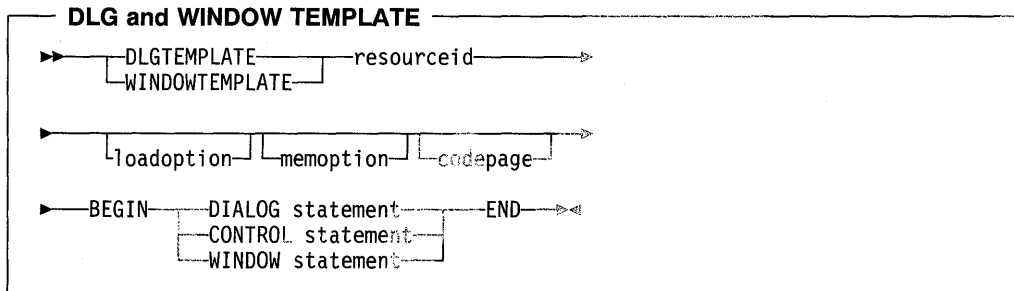
```

## **Dialog and Window Template Statements**

This section describes how to define dialog and window templates.

It also describes the control data and presentation parameter structures that the application needs to create windows and define dialog templates.

DLGTEMPLATE and WINDOWTEMPLATE statements are used by an application to create predefined window and dialog resource templates. These statements are treated identically by the resource compiler and have the following format:



In the following description of the parts of the DLGTEMPLATE and WINDOWTEMPLATE statements, data types are shown after each parameter or option. These are the data types that the parameter or option is converted to when it is compiled.

### **Purpose**

The DLGTEMPLATE or WINDOWTEMPLATE statement marks the beginning of a window template. It defines the name of the window, and its memory and load options.

**resourceid (USHORT)**

Either a unique name or an integer number identifying the resource.

**loadoption (LOADOPTION)**

The default is LOADONCALL.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

**memoption (MEMOPTION)**

The default is MOVEABLE.

See "Resource Load and Memory Options" on page 31-2 for a description of *MEMOPTION*.

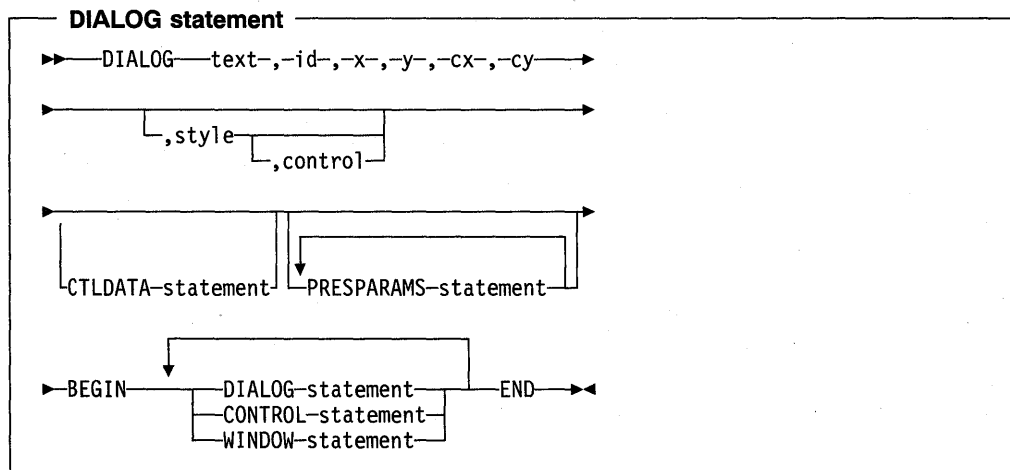
**code page (USHORT)**

The code page of the text in the template.

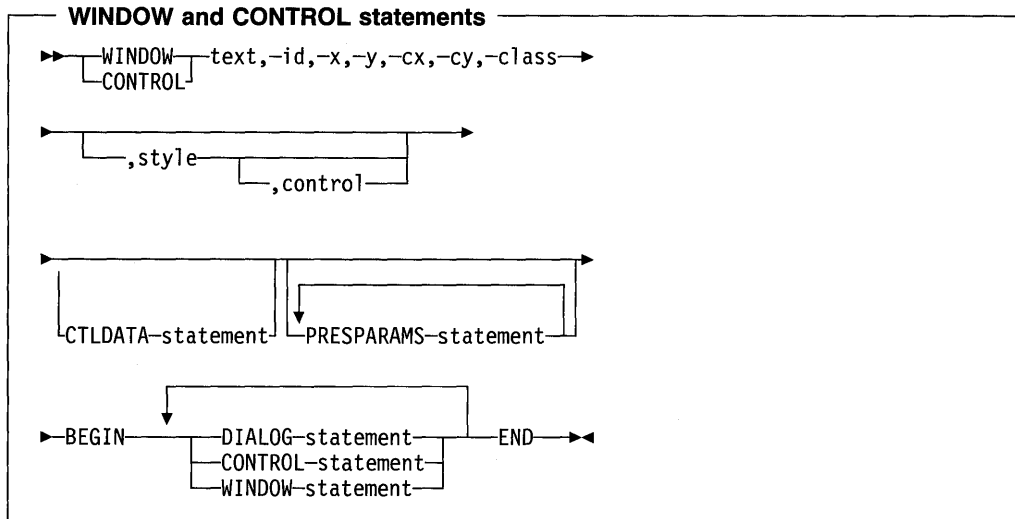
Alternatively, (l) can be used in place of BEGIN and (j) in place of END.

The DLGTEMPLATE and WINDOWTEMPLATE keywords are synonymous.

The DIALOG statement defines a dialog-box window that can be created by an application and has the following format:



The WINDOW and CONTROL statements have the format:



**Note:** The WINDOW and CONTROL keywords are synonymous.

The DIALOG, CONTROL, and WINDOW statements between the BEGIN and END statements are defined as child windows. Presentation parameters always apply to the whole control. They cannot be changed for the individual items within the control.

Following is the description of the parameters for these statements.

### Purpose

These statements mark the beginning of a window. They define the starting location on the display screen, its width, its height, and other details such as style.

**Note:** Not all values may be specified for each statement type. For details, see the call syntax diagrams.

### text (PCH)

A string, enclosed in double quotes, that is displayed in the title-bar control, if it exists. To insert a double-quote character (") in the text, use two double-quote characters ("").

### id (USHORT)

Item identifier.

### x,y (SHORT)

Integer numbers specifying the x- and y-coordinates on the display screen of the lower left corner of the dialog. X and y are in dialog coordinates. The exact meaning of the coordinates depends on the style defined by the style argument. For normal dialogs, the coordinates are relative to the origin of the parent window. For FCF\_SCREENALIGN style boxes, the coordinates are relative to the origin of the display screen. With FCF\_MOUSEALIGN, the coordinates are relative to the position of the pointer at the time the dialog is created.



**cx,cy (SHORT)**

Integer numbers specifying the width and height of the window.

**class (PCH)**

The class of the window or control to be created.

**Note:** For a DIALOG statement the class is fixed as WC\_FRAME and cannot be specified.

**style (ULONG)**

Any additional window style, frame style, or other class-specific style.

The default style is WS\_SYNCPAINT | WS\_CLIPSIBLINGS | WS\_SAVEBITS | FS\_DLGBORDER. If the FS\_DLGBORDER or WS\_SAVEBITS styles are not required, they should be preceded by the keyword "NOT." For example:

```
NOT FS_DLGBORDER | FS_BORDER | NOT WS_SAVEBITS
```

replaces the FS\_DLGBORDER default style by the FS\_BORDER style and removes the WS\_SAVEBITS style. Note that the logic of the NOT keyword is different from the corresponding operator in the C language.

It is not possible to remove the default WS\_SYNCPAINT and WS\_CLIPSIBLINGS styles.

**control (ULONG)**

Frame Creation Flags (FCF\_\*; see page 13-1) for the window

This data is placed in the control data field in the correct format for a window of class WC\_FRAME.

**Note:** FCF\_SHELLPOSITION has no effect if specified in a template.

**CTLDATA Statement**

A statement used to define control data for the control. For more information on this statement, see "Control Data Statement" on page 31-28

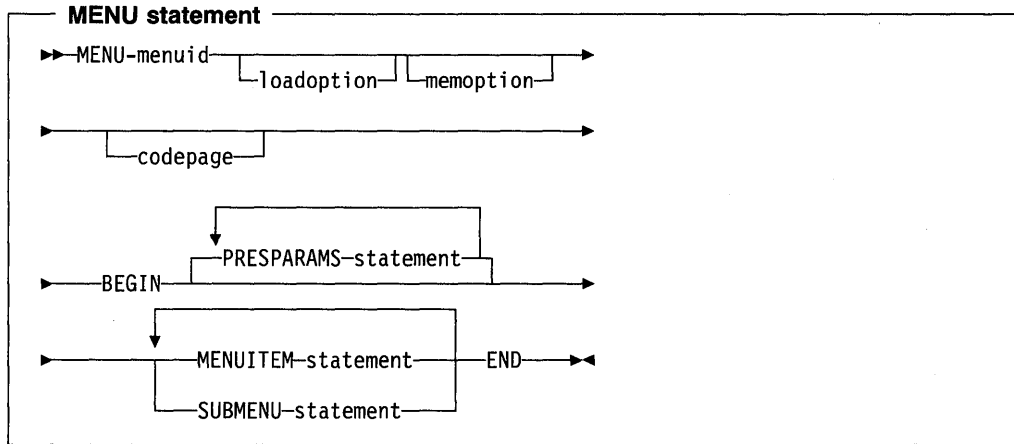
**PRESPARAMS Statement**

A statement used to define presentation parameters. For more information on this statement, see "Presentation Parameters Statement" on page 31-28

**MENU Statement**

The MENU statement defines the contents of a menu resource. A menu resource is a collection of information that defines the appearance and function of an application menu. A menu can be used to create an action bar.

The MENU statement has the form:



**menuid (USHORT)**

A name or number used to identify the menu resource.

**loadoption (LOADOPTION)**

The default is LOADONCALL.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

**memoption (MEMOPTION)**

The default is MOVEABLE.

See "Resource Load and Memory Options" on page 31-2 for a description of *MEMOPTION*.

**codepage (USHORT)**

The code page of the text.

**PRESPARAMS statement**

A special resource statement used to define presentation parameters. These are discussed in more detail in "Presentation Parameters Statement" on page 31-28.

**MENUITEM statement**

A special resource statement used to define the items in the menu. These are discussed in more detail in "Menu Item Statements" on page 31-18.

**SUBMENU statement**

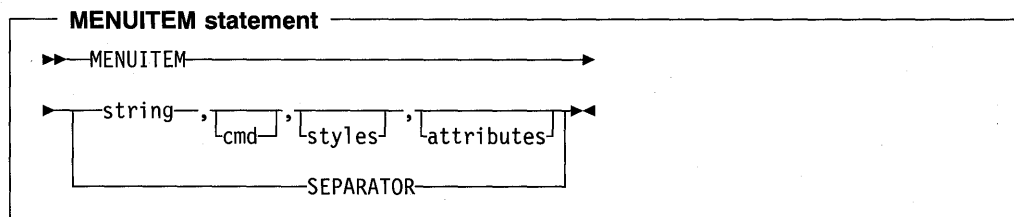
A special resource statement used to define a submenu. SUBMENU statements are discussed in more detail in "Submenu Statements" on page 31-19.

**Example:** Following is an example of a complete MENU statement:

```
MENU sample
BEGIN
MENUITEM "~Alpha", 100, MIS_TEXT
SUBMENU "~Beta", 101, MIS_TEXT
BEGIN
MENUITEM "~Green", 200, MIS_TEXT
MENUITEM "~Blue", 201, MIS_TEXT,MIA_CHECKED
END
END
```

**Menu Item Statements:** MENUITEM statements are used in the item-definition section of a MENU statement to define the names and attributes of the actual menu items. Any number of statements can be given; each defines a unique item. The order of the statements defines the order of the menu items.

**Note:** The MENUITEM statements can only be used within an item-definition section of a MENU statement.



### **string (PCH)**

A string, enclosed in double quotation marks, specifying the text of the menu item.

To insert a double-quote character (") in the text, use two double-quote characters (").

If the **styles** parameter does not contain MIS\_TEXT, the string is ignored but must still be specified. An empty string ("") should be specified in this instance.

To indicate the mnemonic for each item, insert the tilde character (~) in the string preceding the mnemonic character.

For MENUITEM statements within a SUBMENU (that is, pull-down menus) text may be split into a second column with an alignment substring. To right-align items insert "a" in the text where alignment should begin. To left-align a second column of text insert "\t" in the text where alignment should begin. For each SUBMENU the longest item in the second column determines the width of that column. Only one alignment substring should be used in a menu item.

### **cmd (USHORT)**

The value of the WM\_COMMAND, WM\_HELP, or WM\_SYSCOMMAND message generated by the item when it is selected. It identifies the selection made and should be unique within one menu definition.

**styles (USHORT)**

One or more menu options defined by the MIS\_\* constants, ORed together with the "|" operator. For definitions of the MIS\_\* constants, see "Menu Item Styles" on page 15-2.

**attributes (USHORT)**

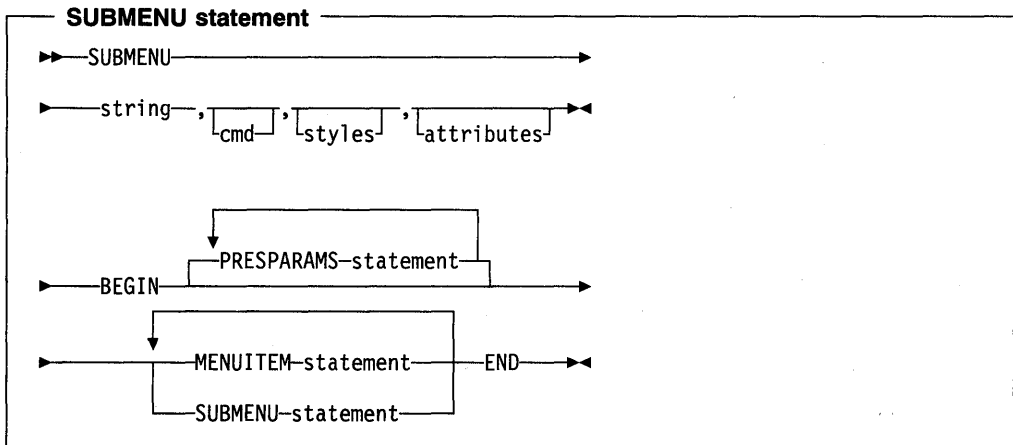
One or more menu options defined by the MIA\_\* constants, ORed together with the "|" operator. For definitions of the MIA\_\* constants, see "Menu Item Attributes" on page 15-3.

The style MIS\_SUBMENU must not be used with this statement. See "Submenu Statements" for the SUBMENU statement.

**Examples:**

```
MENUITEM "Alpha", 1, MIS_TEXT,MIA_ENABLED|MIA_CHECKED,'A'
MENUITEM "Beta", 2, MIS_TEXT,, 'B'
```

**Submenu Statements:** In addition to simple items, a menu definition can contain the definition of a submenu. A submenu can itself invoke a lower level submenu.



**string (PCH)**

A string, enclosed in double quotation marks, specifying the text of the menu item.

To insert a double-quote character (") in the text, use two double-quote characters ("").

If the **styles** parameter does not contain MIS\_TEXT, the string is ignored but must still be specified. An empty string ("" ) should be specified in this instance.

**cmd (USHORT)**

The value of the WM\_COMMAND, WM\_HELP, or WM\_SYSCOMMAND message generated by the item when it is selected. It identifies the selection made and should be unique within one menu definition.

### **styles (USHORT)**

One or more menu options defined by the MIS\_ constants, ORed together with the “|” operator.

In the SUBMENU statement, the style MIS\_SUBMENU is always ORed with the styles given. If no value is supplied, the default value of MIS\_TEXT and MIS\_SUBMENU is used.

### **attributes (USHORT)**

One or more menu options defined by the MIA\_ constants, ORed together with the | operator.

### **Example:**

```
MENU chem
BEGIN

SUBMENU "~Elements", 2, MIS_TEXT
BEGIN
    MENUITEM "~Oxygen", 200, MIS_TEXT
    MENUITEM "~Carbon", 201, MIS_TEXT,MIA_CHECKED
    MENUITEM "~Hydrogen", 202, MIS_TEXT
END

SUBMENU "~Compounds", 3, MIS_TEXT
BEGIN
    MENUITEM "~Glucose", 301, MIS_TEXT
    MENUITEM "~Sucrose", 302, MIS_TEXT,MIA_CHECKED
    MENUITEM "~Lactose", 303, MIS_TEXT|MIS_BREAK
    MENUITEM "~Fructose", 304, MIS_TEXT
END

END
```

**SEPARATOR Menu Item:** There is a special form of the MENUITEM statement that is used to create a horizontal dividing bar between two active menu items in a pull-down menu. The SEPARATOR menu item is itself inactive and has no text associated with it nor a **cmd** value.

### **Example**

```
MENUITEM "~Roman", 206, MIS_TEXT
MENUITEM SEPARATOR
MENUITEM "20 ~Point", 301, MIS_TEXT
```

**Menu Template:** Menu templates are data structures used to define menus. Menu templates can be loaded as resources or created dynamically, or embedded in dialog templates, which in turn can be loaded as resources or created dynamically. Templates loaded as resources cannot contain references to bit maps or owner-drawn items. A menu template consists of a sequence of variable-length records. Each record in a menu template

defines a menu item. If a menu item contains a reference to a submenu, the menu template that defines that submenu is placed after the definition of that particular menu item.

**Template Format:** A menu template has the following format:

**Length (USHORT)**

The length of the menu template.

**Version (USHORT)**

The template version. Versions 0 and 1 are valid.

**Code page (USHORT)**

The identifier of the code page used for the text items within the menu (but not any submenus, which each have their own code pages).

**Item offset (USHORT)**

The offset of the items from the start of the template, in bytes.

**Count (USHORT)**

The count of menu items.

**Presentation parameters offset (USHORT)**

Offset of presentation parameters from the start of the template, in bytes. This field is only present for version 1 of the template.

**Menu Items**

A variable-sized array of menu items as follows:

**Style (USHORT)**

Menu item styles (MIS\_\*; see page 15-2) combined with the logical-OR operator.

**Attributes (USHORT)**

Menu item attributes (MIA\_\*; see page 15-3) combined with the logical-OR operator.

**Item (USHORT)**

An application-provided identifier for the menu item.

**Variable data**

Following the identifier is a variable data structure whose format depends upon the value of **Style**:

**MIS\_TEXT**

**Text (PSZ)**

Null-terminated text string.

**MIS\_SUBMENU**

A menu template structure.

**MIS\_BITMAP**

**Text (PCH)**

Null-terminated text string.

For MIS\_BITMAP menu items, the item text string can be used to derive the resource identifier from which a bit map is loaded. There are three instances:

- The first byte is null; that is, no resource is defined and it is assumed that the application subsequently provides a bit-map handle for the item.
- The first byte is 0xFF, the second byte is the low byte of the resource identifier, and the third byte is the high byte of the resource identifier.
- The first character is "#," and subsequent characters make up the decimal text representation of the resource identifier.

The resource is assumed to reside in the resource file of the current process.

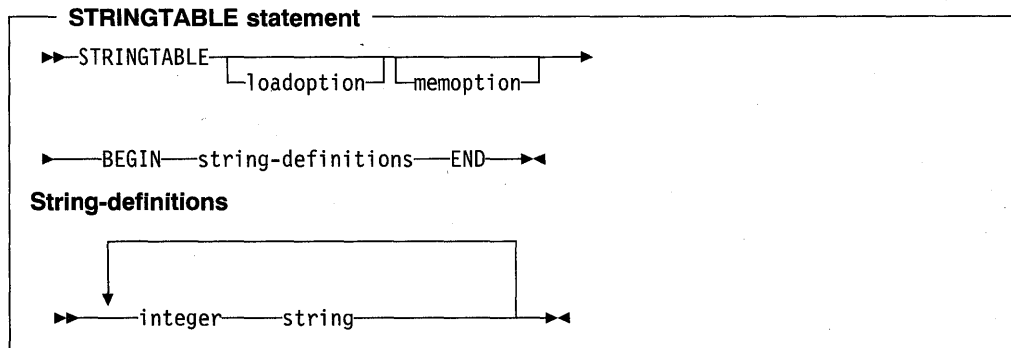
If the string is empty or does not follow the format above, no resource is loaded.

### STRINGTABLE Statement

The STRINGTABLE statement defines one or more string resources for an application. String resources are null-terminated ASCII strings that can be loaded, when needed, from the executable file, using the WinLoadString function.

**Note:** The ASCII strings can include no more than 256 characters, including the NULL termination character.

The STRINGTABLE statement has the form:



#### loadoption (LOADOPTION)

An optional keyword specifying when the resource is to be loaded. It must be one of:

- |                   |                                 |
|-------------------|---------------------------------|
| <b>PRELOAD</b>    | Resource is loaded immediately. |
| <b>LOADONCALL</b> | Resource is loaded when called. |

The default is LOADONCALL.

See "Resource Load and Memory Options" on page 31-2 for a description of *LOADOPTION*.

### **memoption (MEMOPTION)**

Consists of the following keyword or keywords, specifying whether the resource is fixed or movable and whether it is discardable:

|                    |   |
|--------------------|---|
| <b>FIXED</b>       | Resource remains at a fixed memory location.          |
| <b>MOVEABLE</b>    | Resource can be moved if necessary to compact memory. |
| <b>DISCARDABLE</b> | Resource can be discarded if no longer needed.        |

The default is MOVEABLE and DISCARDABLE.

See "Resource Load and Memory Options" on page 31-2 for a description of MEMOPTION.

### **string (PCH)**

A string, enclosed in double quotation marks. To insert a double-quote character (") in the text, use two double-quote characters ("").

**Note:** A string may be defined on more than one line if each line begins and ends with a double-quote. If newline characters are desired after each line, there should be a double-quote at the beginning of the first line and at the end of the last line only.

The string may contain any ASCII characters. Because (\) is interpreted as an escape character, use (\\) to generate a (\).

The following escape sequences may be used:

| <b>Escape Sequence</b> | <b>Name</b>                    |
|------------------------|--------------------------------|
| \\t                    | Horizontal tab                 |
| \\a                    | Bell (alert)                   |
| \\nnn                  | ASCII character (octal)        |
| \\xdd                  | ASCII character (hexadecimal). |

The sequences \\ddd and \\xdd allow any character in the ASCII character set to be inserted in the character string. Thus, the horizontal tab could be entered as \\X09, \\011 or \\t.

### **Example**

```
#define IDS_STRING1 1
#define IDS_STRING2 2
#define IDS_STRING3 3

STRINGTABLE
BEGIN
    IDS_STRING1, "The first two strings in this table are identical."
    IDS_STRING2, "The first two strings "
    "in this table are identical."
    IDS_STRING3, "This string will contain a newline character
    before it continues on this line."
END
```



---

## Templates, Control Data, and Presentation Parameters

### Dialog Template

A dialog template is a data structure used to define a dialog box. Dialog templates can be loaded from resources or created dynamically in memory. Dialog templates define windows of any window class that contain child windows of any class. For standard dialog windows, the dialog window itself is created with the `WC_FRAME` class, and its children are any of the preregistered control classes.

The dialog template specifies all the information required to create a dialog box and its children.

### Dialog Coordinates

Coordinates in a dialog template are specified in *dialog coordinates*. These are based on the default character cell size; a unit in the horizontal direction is 1/4 the default character-cell width, and a unit in the vertical direction is 1/8 the default character-cell height. The origin is the bottom left-hand corner of the dialog box.

### Dialog Template Format and Contents

A dialog template has these sections:

|                  |  |
|------------------|--|
| <b>Header</b>    | Defines the type of template format and contains information about the location of the other sections of the template. It also contains a summary of the status of the individual controls contained within the dialog box.  |
| <b>Items</b>     | Defines each of the controls that comprise the dialog box.   |
| <b>Data area</b> | Contains the data values associated with each control. Each control defined in the item section contains pointers to the data area section. The data area also contains presentation parameter definitions. The data area is not necessarily a contiguous portion of the template. User data can be placed anywhere in the template if it does not interfere with other defined information. |

The sections of a dialog template are illustrated in Figure 31-1 on page 31-25.

#### Notes:

1. Throughout the dialog template all lengths are in bytes. String lengths do not include any null terminator that may be present. When strings are passed to the Presentation Interface, the length specifications are used and any null terminators are ignored. When strings are returned by the Presentation Interface, length specifications and null terminators are both supplied; therefore, space must be allowed for a null terminator.
2. All offsets are in bytes from the start of the dialog template structure.

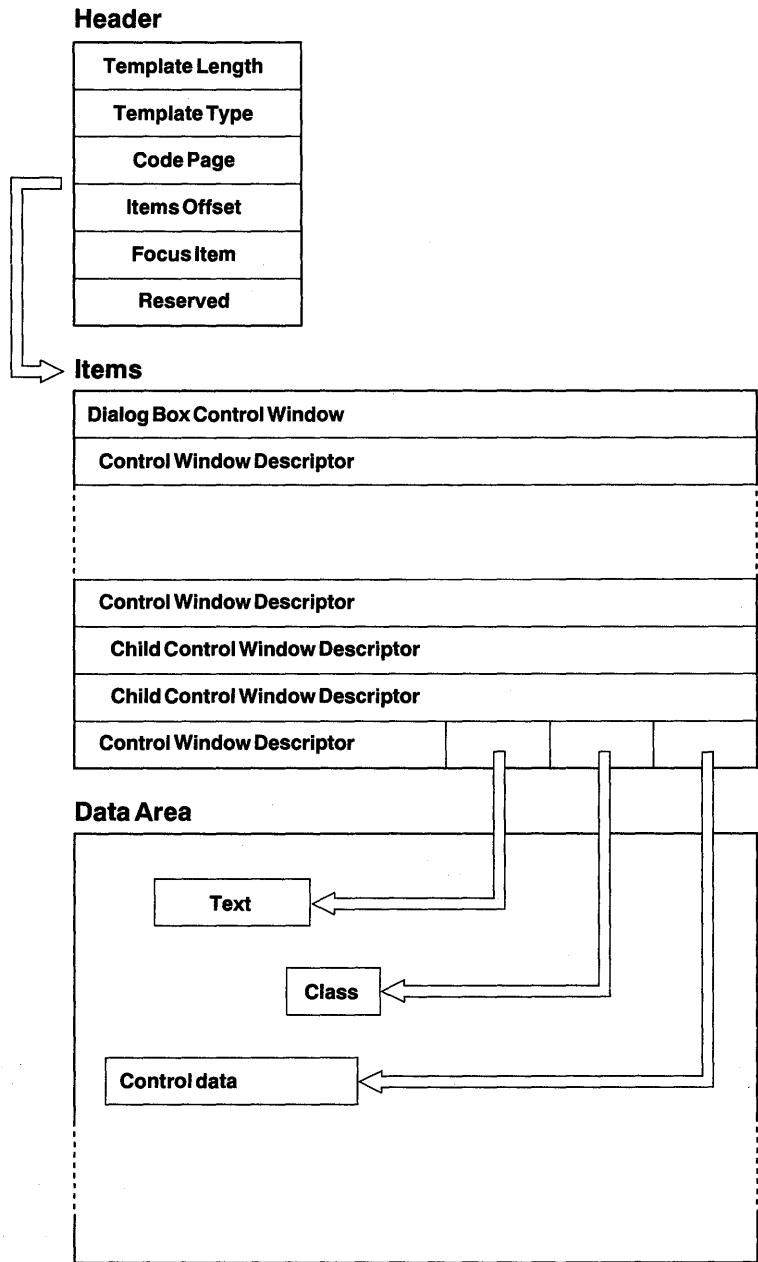


Figure 31-1. Dialog Template

## Header

The dialog template header consists of:

**Template length** (*USHORT*)

The overall length of the dialog template.

**Template type** (*USHORT*)

The dialog template format type. The format defined is type 0.

**Code page** (*USHORT*)

The code page of the text in the dialog template.

**Items offset** (*USHORT*)

The offset of the array of dialog items.

**Reserved** (*USHORT*)

Must be 0.

**Focus item** (*USHORT*)

The index in the array of dialog items of the control to receive the focus. If this value is 0, or if the identified control cannot receive the focus, for example because it is a static control, the focus is passed to the first item within the template that can receive the focus.

**Reserved** (*USHORT*)

Must be 0.

## Items

The dialog template items are specified as elements of an array that also defines the hierarchy of the control windows of the dialog box. Each element of the array is a control window descriptor and defines some control or a child of some control, so that every control within the dialog box is described by this array. The first descriptor is the specification of the dialog box itself.

The dialog template items consist of:

**Reserved** (*USHORT*) (*BOOL16*)

Must be 0.

**Children** (*USHORT*)

The number of dialog item child windows that are owned by this dialog item.

This is the number of elements following in the array that are created as child windows of this window. Each window can have any number of child windows, which allows for a tree-structured arrangement.

For example, in Figure 31-1 on page 31-25, assuming that there are no more dialog items than are shown, the first item, the dialog box control window descriptor, has three children. The second item has no children, the third item has two children, and the remaining three items have no children.

**Class name length (USHORT)**

The length of the window class name string.

**Class name offset (USHORT)**

The offset of the window class name string.

**Text length (USHORT)**

The length of the text string.

For controls that allow input of text, this is the current text length, not the maximum text length, and so this value changes when text is put into the control.

**Text offset (USHORT)**

The offset of the text string.

**Style (ULONG) (BOOL32)**

The window style of the control.

The standard style bits are 16 bits. The use of the remaining 16 bits depends on the class of the control.

**x (SHORT)****y (SHORT)**

The position of the origin of the dialog item. This is specified in dialog coordinates, with x and y relative to the origin of the parent window.

**cx (SHORT)****cy (SHORT)**

The size of the dialog item in dialog coordinates; it must be greater than 0.

**Identifier (USHORT)**

An application-defined identifier for the dialog item.

**Reserved (USHORT)**

Must be zero.

**Control data offset (USHORT)**

The offset of the control-specific data for this dialog item. A value of 0 indicates that there is no control data for this dialog item.

## Data Area

The dialog template data area contains the following different types of objects: **text**, **class name**, **presentation parameters**, and **control data**. These objects can be placed anywhere within the data area. They do not have to be in contiguous storage, and so an application can place data for its own use between these objects.

The dialog template data area contains:

**Text (PCH)**

The textual data associated with a dialog item.

**Class name (PCH)**

The name of the window class.

### Presentation parameters (*PRESPARAMS*)

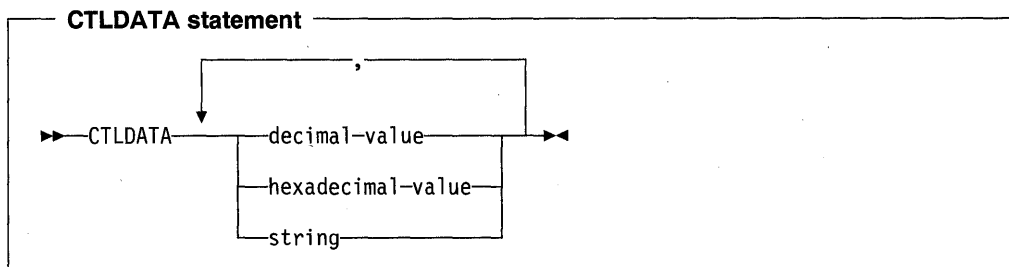
Presentation parameters are defined in "Presentation Parameters Statement" on page 31-28.

### Control data (*CTLDATA*)

For more information, see "Control Data Statement."

## Control Data Statement

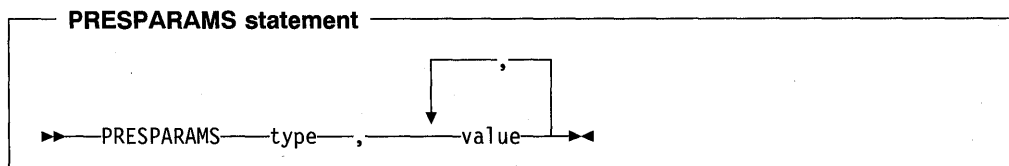
The optional *CTLDATA* statement is used to define control data for the control. Hexadecimal or decimal word constants follow the *CTLDATA* statement, separated with commas.



In addition to hexadecimal or decimal data, the *CTLDATA* statement can be followed by the *MENU* keyword, followed by a menu template in a *BEGIN/END* block. This creates a menu template as the control data of the window.

## Presentation Parameters Statement

The optional *PRESPARAMS* statement is used to define presentation parameters. The syntax of the *PRESPARAMS* statement is as follows.



A presentation parameter consists of:

### **type** (*ULONG*)

The presentation parameter attribute type. See the *PARAM* data type for a description of valid types.

A string can be used to specify the type for a user type. If this is done, the string type is converted into a string atom when the dialog template is read into memory. Thereafter, this presentation parameter is referred to by this string atom. The application can use the atom manager API to match the string and the string atom.

### **value (LONG or PSZ)**

One or more values depending upon the attribute type.

If the value is enclosed in quotes it is a zero-terminated string. Otherwise, it is converted to a LONG. There may be more than one value, depending upon the type. See PARAM data type for a description of the values required for system-defined presentation parameters.

**Examples:** The following are examples of PRESPARAMS statements:

```
PRESPARAMS PP_BORDERCOLOR, 0x00ff00ffL
PRESPARAMS PP_FONTNAME, "12.Helv"
PRESPARAMS "my color", 0x00ff00ffL
PRESPARAMS "my param", 0, 1, 2, 3, "Hi there"
```

## **Parent/Child/Owner Relationship**

The format of the DLGTEMPLATE and WINDOWTEMPLATE resources is very general to allow tree-structured relationships within the resource format. The general layout of the templates is:

```
WINDOWTEMPLATE id
BEGIN
    WINDOW winTop           the top-level window
    BEGIN
        WINDOW wind1
        WINDOW wind2
        WINDOW wind3
        BEGIN
            WINDOW wind4
        END
        WINDOW wind5
    END
END
```

In this example, the top-level window is identified by **winTop**. It has four child windows: **wind1**, **wind2**, **wind3**, and **wind5**. **wind3** has one child window, **wind4**. When each of these windows is created, the parent and the owner are set to be the same.

The only time when the parent and owner windows are not the same is when frame controls are automatically created by a frame window.

Note that the WINDOW statements in the example above could also have been CONTROL or DIALOG statements.

## Predefined Window Classes

The CONTROL statement can be used to define a window control of any class. Window classes may be user defined or one of a predefined set provided by the operating system. The following classes are provided in the OS/2 operating system.

|               |  |
|---------------|--|
| WC_FRAME      | Application frame control.                     |
| WC_STATIC     | Text and group boxes.                          |
| WC_BUTTON     | Push button, check box or radio button.        |
| WC_COMBOBOX   | Combination of an entry field and list box.    |
| WC_ENTRYFIELD | Single line entry field.                       |
| WC_MLE        | Multiple line entry field.                     |
| WC_LISTBOX    | List box.                                      |
| WC_MENU       | Application action bar, menus and popup menus. |
| WC_SCROLLBAR  | Horizontal or vertical scroll bar.             |
| WC_TITLEBAR   | Application title bar.                         |
| WC_SPINBUTTON | Spin button entry field.                       |
| WC_CONTAINER  | Container list.                                |
| WC_SLIDER     | Horizontal or vertical slider control.         |
| WC_VALUESET   | Value set control.                             |
| WC_NOTEBOOK   | Notebook control.                              |

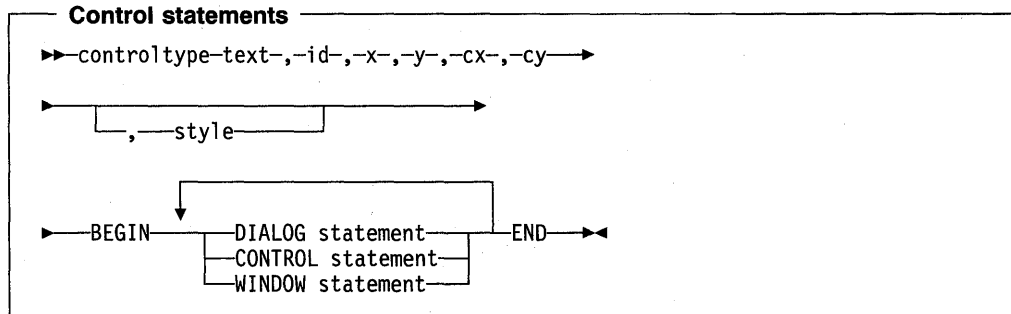
These controls make up the standard user interface components for applications. The following example shows a simple listbox control.

```
CONTROL "", 1, 10, 20, 60, 40, WC_LISTBOX, WS_VISIBLE
```

## Predefined Control Statements

In addition to the general form of the CONTROL statement, there are special control statements for commonly used controls. These statements define the attributes of the child control windows that appear in the window.

Control statements have this general form:



The following six controls are exceptions to this form because they do not take a text field. See the LISTBOX control statement for the form of these six controls.

- CONTAINER
- LISTBOX
- NOTEBOOK
- SLIDER
- SPINBUTTON
- VALUESET

### **controltype**

is one of the keywords described below, defining the type of the control.

### **text (PCH)**

is a string specifying the text to be displayed. The string must be enclosed in double quotation marks. The manner in which the text is displayed depends on the particular control, as detailed below.

To indicate the mnemonic for each item, insert the tilde character (~) in the string preceding the mnemonic character.

The double quotation marks are required for the COMBOBOX title even if no title is used.

### **id (USHORT)**

is a unique integer number identifying the control.

### **x,y (SHORT)**

are integer numbers specifying the x- and y-coordinates of the lower left corner of the control, in dialog coordinates. The coordinates are relative to the origin of the dialog.

### **cx,cy (SHORT)**

are integer numbers specifying the width and height of the control.

The x, y, cx, and cy fields can use addition and subtraction operators (+ and -). For example, 15 + 6 can be used for the x-field.

Styles can be combined using the (|) operator.

The control type keywords are shown below, with their classes and default styles:

### **AUTOCHECKBOX**

|               |   |
|---------------|---|
| Class         | WC_BUTTON                               |
| Default style | WS_TABSTOP, WS_VISIBLE, BS_AUTOCHECKBOX |

### **AUTORADIOBUTTON**

|               |  |
|---------------|--|
| Class         | WC_BUTTON                                  |
| Default style | BS_AUTORADIOBUTTON, WS_TABSTOP, WS_VISIBLE |

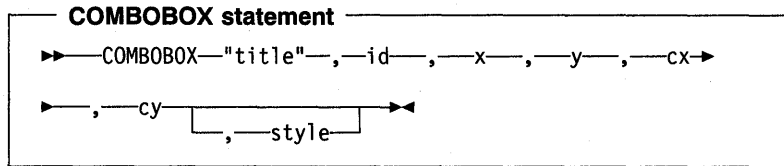
### **CHECKBOX**

|               |                                     |
|---------------|-------------------------------------|
| Class         | WC_BUTTON                           |
| Default style | BS_CHECKBOX, WS_TABSTOP, WS_VISIBLE |



## COMBOBOX

**Format** The form of the COMBOBOX control statement is shown below.  
The fields have the same meaning as in the other control statements.



**Class** WC\_COMBOBOX  
**Default style** CBS\_SIMPLE, WS\_TABSTOP, WS\_VISIBLE

## CONTAINER

**Format** The CONTAINER control statement does not contain a text field, so it has the same format as the LISTBOX statement.

**Class** WC\_CONTAINER  
**Default style** WS\_TABSTOP, WS\_VISIBLE, CCS\_SINGLESEL

## CTEXT

**Class** WC\_STATIC  
**Default style** SS\_TEXT, DT\_CENTER, WS\_GROUP, WS\_VISIBLE

## DEFPUSHBUTTON

**Class** WC\_BUTTON  
**Default style** BS\_DEFAULT, BS\_PUSHBUTTON, WS\_TABSTOP, WS\_VISIBLE

## EDITTEXT

**Class** WC\_ENTRYFIELD  
**Default style** WS\_ENTRYFIELD, WS\_TABSTOP, WS\_VISIBLE, ES\_AUTOSCROLL

## ENTRYFIELD

**Class** WC\_ENTRYFIELD  
**Default style** WS\_TABSTOP, ES\_LEFT, WS\_VISIBLE

## FRAME

**Class** WC\_FRAME  
**Default style** WS\_VISIBLE

## GROUPBOX

**Class** WC\_STATIC  
**Default style** SS\_GROUPBOX, WS\_TABSTOP, WS\_VISIBLE

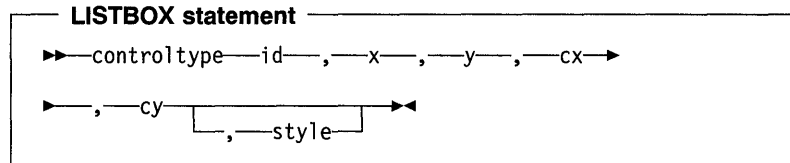
## ICON

**Class** WC\_STATIC  
**Default style** SS\_ICON, WS\_VISIBLE

## LISTBOX

Format

The form of the LISTBOX control statement is different from the general form because it does not take a text field, however the fields have the same meaning as in the other control statements. The form of the LISTBOX control statement is shown below.



Class

WC\_LISTBOX

Default style

LBS\_NOTIFY, LBS\_SORT, WS\_VSCROLL, WS\_BORDER,  
WS\_VISIBLE

## LTEXT

Class

WC\_STATIC

Default style

SS\_TEXT, DT\_LEFT, WS\_GROUP, WS\_VISIBLE

## MLE

Class

WC\_MLE

Default style

WS\_MLE, WS\_TABSTOP, WS\_VISIBLE, MLS\_BORDER

## NOTEBOOK

Format

The NOTEBOOK control statement does not contain a text field, so it has the same format as the LISTBOX statement.

Class

WC\_NOTEBOOK

Default style

WS\_NOTEBOOK, WS\_TABSTOP, WS\_VISIBLE

## PUSHBUTTON

Class

WC\_BUTTON

Default style

BS\_PUSHBUTTON, WS\_TABSTOP, WS\_VISIBLE

## RADIOBUTTON

Class

WC\_BUTTON

Default style

BS\_RADIOBUTTON, WS\_TABSTOP, WS\_VISIBLE

## RTEXT

Class

WC\_STATIC

Default style

SS\_TEXT, DT\_RIGHT, WS\_GROUP, WS\_VISIBLE

## SLIDER

Format

The SLIDER control statement does not contain a text field, so it has the same format as the LISTBOX statement.

Class

WC\_SLIDER

Default style

WS\_SLIDER, WS\_TABSTOP, WS\_VISIBLE

## SPINBUTTON

Format            The SPINBUTTON control statement does not contain a text field, so it has the same format as the LISTBOX statement.

Class             WC\_SPINBUTTON  
Default style    WS\_TABSTOP, WS\_VISIBLE, SPBS\_MASTER

## VALUESET

Format            The VALUESET control statement does not contain a text field, so it has the same format as the LISTBOX statement.

Class             WC\_VALUESET  
Default style    WS\_VALUESET, WS\_TABSTOP, WS\_VISIBLE

**Examples:** The following is a complete example of a DIALOG statement:

```
DLGTEMPLATE errmess
BEGIN
    DIALOG "Disk Error", 100, 10, 10, 300, 110
    BEGIN
        CTEXT "Select One:", 1, 10, 80, 280, 12
        RADIOBUTTON "Retry", 2, 75, 50, 60, 12
        RADIOBUTTON "Abort", 3, 75, 30, 60, 12
        RADIOBUTTON "Ignore", 4, 75, 10, 60, 12
    END
END
```

This is an example of a WINDOWTEMPLATE statement that is used to define a specific kind of window frame. Calling Load Dialog with this resource automatically creates the frame window, the frame controls, and the client window (of class MyClientClass).

```
WINDOWTEMPLATE wind1
BEGIN
    FRAME "My Window", 1, 10, 10, 320, 130, WS_VISIBLE,
        FCF_STANDARD | FCF_VERTSCROLL
    BEGIN
        WINDOW "", FID_CLIENT, 0, 0, 0, 0, "MyClientClass",
            style
    END
END
```

This example creates a resource template for a parallel dialog identified by the constant **parallel1**. It includes a frame with a title bar, a system menu, and a dialog-style border. The parallel dialog has three auto radio buttons in it.

```
DLGTEMPLATE parallel1
BEGIN
    DIALOG "Parallel Dialog", 1, 50, 50, 180, 110
        CTLDATA FCF_TITLEBAR | FCF_SYSMENU | FCF_DLGBOARDER
        BEGIN
            AUTORADIOBUTTON "Retry", 2, 75, 80, 60, 12
            AUTORADIOBUTTON "Abort", 3, 75, 50, 60, 12
            AUTORADIOBUTTON "Ignore", 4, 75, 30, 60, 12
        END
    END
END
```

---

## Resource (.RES) File Specification

The format for the .RES file is:

```
(/TYPE NAME FLAGS SIZE BYTES/)+
```

Where:

**TYPE** is either a null-terminated string or an ordinal, in which instance the first byte is 0xFF followed by an INT that is the ordinal.

```
/* Predefined resource types */
#define RT_POINTER      1
#define RT_BITMAP      2
#define RT_MENU        3
#define RT_DIALOG      4
#define RT_STRING      5
#define RT_FONTDIR     6
#define RT_FONT        7
#define RT_ACCELTABLE  8
#define RT_RCDATA      9
#define RT_DLGINCLUDE  11
#define RT_FKALONG     17
#define RT_HELPTABLE   18
```

**NAME** is the same format as TYPE. There are no predefined names.

**FLAGS** is an unsigned value containing the memory manager flags:

```
#define NSTYPE      0x0007 /* Segment type mask */
#define NSCODE     0x0000 /* Code segment */
#define NSDATA     0x0001 /* Data segment */
#define NSITER     0x0008 /* Iterated segment flag */
#define NSMOVE     0x0010 /* Moveable segment flag */
#define NSPURE     0x0020 /* Pure segment flag */
#define NSPRELOAD  0x0040 /* Preload segment flag */
#define NSEXRD     0x0080 /* Execute-only (code segment),
                          /* or read-only (data segment) */
#define NSRELOC    0x0100 /* Segment has relocations */
#define NSCONFORM  0x0200 /* Segment has debug info */
#define NSDPL      0x0C00 /* 286 DPL bits */
#define NSDISCARD  0x1000 /* Discard bit for segment */
#define NS32BIT    0x2000 /* 32-BIT code segment */
#define NSHUGE     0x4000 /* Huge memory segment */
```

**SIZE** is a LONG value defining how many bytes follow in the resource.

**BYTES** is the stream of bytes that makes up the resource.

Any number of resources can appear one after another in the .RES file.

---

## Chapter 32. Code Pages

The initialization file contains country information relating to date, time, and numeric formats. It does not contain code-page information; this is obtained from the CONFIG.SYS file.

Applications start with the default code page. The default code page is set when the operating system is installed. It can be changed subsequently either by reinstalling the operating system or by editing the COUNTRY statement in the CONFIG.SYS file.

A GPI presentation space inherits the code page of the process that created it. The code page changes only when the process calls GpiSetCp

---

### Windowed PM Applications

Windowed PM applications allow the code-page calls to use any of the supported ASCII code pages. These are:

|                      | <b>Char. Set</b> | <b>Code Page</b> |
|----------------------|------------------|------------------|
| Canadian-French      | 993              | 863              |
| Desktop Publishing   | 1146             | 1004             |
| Iceland              | 991              | 861              |
| Latin 1 Multilingual | 980              | 850              |
| Latin 2 Multilingual | 982              | 852              |
| Nordic               | 995              | 865              |
| Portuguese           | 990              | 860              |
| Turkey               | 987              | 857              |
| U.S. (IBM PC)        | 919              | 437              |

Code page 1004 is compatible with Microsoft\*\* Windows\*\*.

The following EBCDIC code pages, based on character set 697, are also available for output:

|                  | <b>Char. Set</b> | <b>Code Page</b> |
|------------------|------------------|------------------|
| Austrian/German  | 697              | 273              |
| Belgian          | 697              | 500              |
| Brazil           | 697              | 037              |
| Czechoslovakia   | 959              | 870              |
| Danish/Norwegian | 697              | 277              |
| Finnish/Swedish  | 697              | 278              |
| French           | 697              | 297              |
| Hungary          | 959              | 870              |
| Iceland          | 697              | 871              |
| International    | 697              | 500              |
| Italian          | 697              | 280              |
| Poland           | 959              | 870              |
| Portuguese       | 697              | 037              |
| Spanish          | 697              | 284              |
| Turkey           | 1152             | 1026             |
| U.K.-English     | 697              | 285              |
| U.S.-English     | 697              | 037              |
| Yugoslavia       | 959              | 870              |

**Note:** Code pages 274 (Belgian) and 282 (Portuguese) can be used to provide access to old data.

The operating system provides the following additional code-page setting and query calls for the supported ASCII and EBCDIC code pages. These calls work independently of the CONFIG.SYS file.

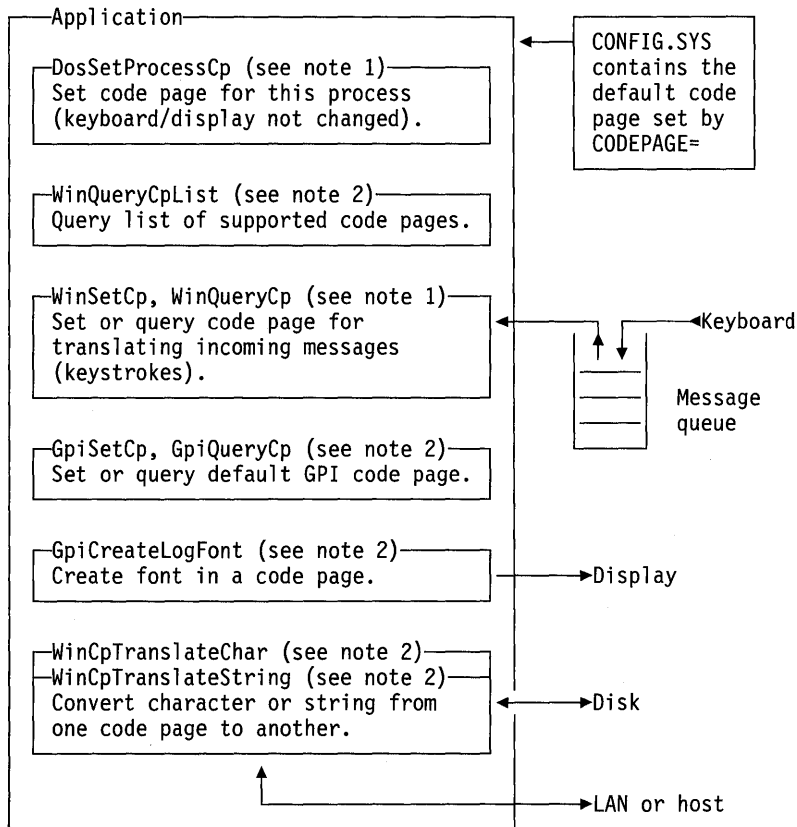
|                         |  |
|-------------------------|--|
| <b>GpiSetCp</b>         | Sets the code page for GPI.                |
| <b>GpiQueryCp</b>       | Queries the code page for GPI.             |
| <b>GpiCreateLogFont</b> | Creates fonts in a code page.              |
| <b>WinSetCp</b>         | Sets the code page for a message queue.    |
| <b>WinQueryCp</b>       | Queries the code page for a message queue. |

WinQueryCpList creates a list of code pages supported by the operating system.

Text entered in a dialog box is supplied to the application in the code page of the queue ('queue code page'). If possible, the code page of a resource (for example, a menu or dialog box) should match the code page of the queue. In general, code page 850 is the best choice for both an application and its resources.

Applications should be able to process data from a variety of sources. Because code page 850 contains most of the characters in other supported code pages, this is usually the best choice for the queue code page.

## OS/2 Code Page Options for PM Applications



Note 1: Either of the two ASCII code pages specified in CONFIG.SYS.  
Code page 1004 is also supported.

Note 2: Any supported ASCII or EBCDIC code page as reported by  
WinQueryCpList.  
Code page 1004 is also supported.

Figure 32-1. OS/2 Code Page Options for PM Applications



---

## OS/2 Font Support for Multiple Code Pages

The operating system supports multiple code pages for text input and output. A single font resource is used to support all the code pages. This section describes the font resource format.

### Font Code-Page Functions

Many of the characters required by each code page are common; for example, the first 128 characters of all the ASCII code pages are identical. This set of characters is called the Universal Glyph List (UGL). A code page is simply a set of pointers into the UGL.

As the characters in every font are in the same order, only one set of code-page translation tables is necessary.

**Note:** The fonts of Microsoft Windows support only code page 1004.

### Font Layout

The following table lists the full character set in the order in which the characters occur in the multi-code-page font. Characters are listed in order of their universal glyph list (UGL) number; the graphic character global identifier (GCGID) and a description of each character are also given.

| UGL | GCGID    | Description                              |
|-----|----------|--|
| 1   | SS000000 | Smiling face                             |
| 2   | SS010000 | Smiling face, reverse image              |
| 3   | SS020000 | Heart suit symbol                        |
| 4   | SS030000 | Diamond suit symbol                      |
| 5   | SS040000 | Club suit symbol                         |
| 6   | SS050000 | Spade suit symbol                        |
| 7   | SM570000 | Bullet                                   |
| 8   | SM570001 | Bullet, reverse image                    |
| 9   | SM750000 | Open circle                              |
| 10  | SM750002 | Open circle, reverse image               |
| 11  | SM280000 | Male symbol                              |
| 12  | SM290000 | Female symbol                            |
| 13  | SM930000 | Musical note                             |
| 14  | SM910000 | Two musical notes                        |
| 15  | SM690000 | Sun symbol                               |
| 16  | SM590000 | Forward arrow indicator                  |
| 17  | SM630000 | Back arrow indicator                     |
| 18  | SM760000 | Up-down arrow                            |
| 19  | SP330000 | Double exclamation point                 |
| 20  | SM250000 | Paragraph symbol (USA)                   |
| 21  | SM240000 | Section symbol (USA), paragraph (Europe) |
| 22  | SM700000 | Solid horizontal rectangle               |
| 23  | SM770000 | Up-down arrow, perpendicular             |
| 24  | SM320000 | Up arrow                                 |
| 25  | SM330000 | Down arrow                               |
| 26  | SM310000 | Right arrow                              |
| 27  | SM300000 | Left arrow                               |
| 28  | SA420000 | Right angle symbol                       |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>                   |
|------------|--------------|--------------------------------------|
| 29         | SM780000     | Left-right arrow                     |
| 30         | SM600000     | Solid triangle                       |
| 31         | SV040000     | Solid triangle, inverted             |
| 32         | SP010000     | Space                                |
| 33         | SP020000     | Exclamation point                    |
| 34         | SP040000     | Quotation marks                      |
| 35         | SM010000     | Number sign                          |
| 36         | SC030000     | Dollar sign                          |
| 37         | SM020000     | Percent sign                         |
| 38         | SM030000     | Ampersand                            |
| 39         | SP050000     | Apostrophe                           |
| 40         | SP060000     | Left parenthesis                     |
| 41         | SP070000     | Right parenthesis                    |
| 42         | SM040000     | Asterisk                             |
| 43         | SA010000     | Plus sign                            |
| 44         | SP080000     | Comma                                |
| 45         | SP100000     | Hyphen/minus sign                    |
| 46         | SP110000     | Period/full stop                     |
| 47         | SP120000     | Slash                                |
| 48         | ND100000     | Zero                                 |
| 49         | ND010000     | One                                  |
| 50         | ND020000     | Two                                  |
| 51         | ND030000     | Three                                |
| 52         | ND040000     | Four                                 |
| 53         | ND050000     | Five                                 |
| 54         | ND060000     | Six                                  |
| 55         | ND070000     | Seven                                |
| 56         | ND080000     | Eight                                |
| 57         | ND090000     | Nine                                 |
| 58         | SP130000     | Colon                                |
| 59         | SP140000     | Semicolon                            |
| 60         | SA030000     | Less than sign/greater than (arabic) |
| 61         | SA040000     | Equal Sign                           |
| 62         | SA050000     | Greater than sign/less than (arabic) |
| 63         | SP150000     | Question mark                        |
| 64         | SM050000     | At sign                              |
| 65         | LA020000     | A capital                            |
| 66         | LB020000     | B capital                            |
| 67         | LC020000     | C capital                            |
| 68         | LD020000     | D capital                            |
| 69         | LE020000     | E capital                            |
| 70         | LF020000     | F capital                            |
| 71         | LG020000     | G capital                            |
| 72         | LH020000     | H capital                            |
| 73         | LI020000     | I capital                            |
| 74         | LJ020000     | J capital                            |
| 75         | LK020000     | K capital                            |
| 76         | LL020000     | L capital                            |
| 77         | LM020000     | M capital                            |
| 78         | LN020000     | N capital                            |
| 79         | LO020000     | O capital                            |
| 80         | LP020000     | P capital                            |
| 81         | LQ020000     | Q capital                            |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>               |
|------------|--------------|----------------------------------|
| 82         | LR020000     | R capital                        |
| 83         | LS020000     | S capital                        |
| 84         | LT020000     | T capital                        |
| 85         | LU020000     | U capital                        |
| 86         | LV020000     | V capital                        |
| 87         | LW020000     | W capital                        |
| 88         | LX020000     | X capital                        |
| 89         | LY020000     | Y capital                        |
| 90         | LZ020000     | Z capital                        |
| 91         | SM060000     | Left bracket                     |
| 92         | SM070000     | Backslash                        |
| 93         | SM080000     | Right bracket                    |
| 94         | SD150000     | Circumflex Accent                |
| 95         | SP090000     | Underline, continuous underscore |
| 96         | SD130000     | Grave accent                     |
| 97         | LA010000     | a small                          |
| 98         | LB010000     | b small                          |
| 99         | LC010000     | c small                          |
| 100        | LD010000     | d small                          |
| 101        | LE010000     | e small                          |
| 102        | LF010000     | f small                          |
| 103        | LG010000     | g small                          |
| 104        | LH010000     | h small                          |
| 105        | LI010000     | i small                          |
| 106        | LJ010000     | j small                          |
| 107        | LK010000     | k small                          |
| 108        | LL010000     | l small                          |
| 109        | LM010000     | m small                          |
| 110        | LN010000     | n small                          |
| 111        | LO010000     | o small                          |
| 112        | LP010000     | p small                          |
| 113        | LQ010000     | q small                          |
| 114        | LR010000     | r small                          |
| 115        | LS010000     | s small                          |
| 116        | LT010000     | t small                          |
| 117        | LU010000     | u small                          |
| 118        | LV010000     | v small                          |
| 119        | LW010000     | w small                          |
| 120        | LX010000     | x small                          |
| 121        | LY010000     | y small                          |
| 122        | LZ010000     | z small                          |
| 123        | SM110000     | Left brace                       |
| 124        | SM130000     | Vertical line, logical OR        |
| 125        | SM140000     | Right brace                      |
| 126        | SD190000     | Tilde                            |
| 127        | SM790000     | House                            |
| 128        | LC420000     | C cedilla capital                |
| 129        | LU170000     | U diaeresis small                |
| 130        | LE110000     | E acute small                    |
| 131        | LA150000     | A circumflex small               |
| 132        | LA170000     | A diaeresis small                |
| 133        | LA130000     | A grave small                    |
| 134        | LA270000     | A overcircle small               |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>              |
|------------|--------------|---------------------------------|
| 135        | LC410000     | C cedilla small                 |
| 136        | LE150000     | E circumflex small              |
| 137        | LE170000     | E diaeresis small               |
| 138        | LE130000     | E grave small                   |
| 139        | LI170000     | I diaeresis small               |
| 140        | LI150000     | I circumflex small              |
| 141        | LI130000     | I grave small                   |
| 142        | LA180000     | A diaeresis capital             |
| 143        | LA280000     | A overcircle capital            |
| 144        | LE120000     | E acute capital                 |
| 145        | LA510000     | AE diphthong small              |
| 146        | LA520000     | AE diphthong capital            |
| 147        | LO150000     | O circumflex small              |
| 148        | LO170000     | O diaeresis small               |
| 149        | LO130000     | O grave small                   |
| 150        | LU150000     | U circumflex small              |
| 151        | LU130000     | U grave small                   |
| 152        | LY170000     | Y diaeresis small               |
| 153        | LO180000     | O diaeresis capital             |
| 154        | LU180000     | U diaeresis capital             |
| 155        | LO610000     | O slash small                   |
| 156        | SC020000     | Pound sterling sign             |
| 157        | LO620000     | O slash capital                 |
| 158        | SA070000     | Multiply sign                   |
| 159        | SC070000     | Florin sign                     |
| 160        | LA110000     | A acute small                   |
| 161        | LI110000     | I acute small                   |
| 162        | LO110000     | O acute small                   |
| 163        | LU110000     | U acute small                   |
| 164        | LN190000     | N tilde small                   |
| 165        | LN200000     | N tilde capital                 |
| 166        | SM210000     | Ordinal indicator, feminine     |
| 167        | SM200000     | Ordinal indicator, masculine    |
| 168        | SP160000     | Question mark, inverted         |
| 169        | SM530000     | Registered trademark symbol     |
| 170        | SM660000     | Logical NOT, end of line symbol |
| 171        | NF010000     | One-half                        |
| 172        | NF040000     | One-quarter                     |
| 173        | SP030000     | Exclamation point, inverted     |
| 174        | SP170000     | Left angled quotes              |
| 175        | SP180000     | Right angled quotes             |
| 176        | SF140000     | Fill character, light           |
| 177        | SF150000     | Fill character, medium          |
| 178        | SF160000     | Fill character, heavy           |
| 179        | SF110000     | Center box bar vertical         |
| 180        | SF090000     | Right middle box side           |
| 181        | LA120000     | A acute capital                 |
| 182        | LA160000     | A circumflex capital            |
| 183        | LA140000     | A grave capital                 |
| 184        | SM520000     | Copyright symbol                |
| 185        | SF230000     | Right box side double           |
| 186        | SF240000     | Center box bar vertical double  |
| 187        | SF250000     | Upper right box corner double   |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>                      |
|------------|--------------|---|
| 188        | SF260000     | Lower right box corner double           |
| 189        | SC040000     | Cent sign                               |
| 190        | SC050000     | Yen sign                                |
| 191        | SF030000     | Upper right box corner                  |
| 192        | SF020000     | Lower left box corner                   |
| 193        | SF070000     | Middle box bottom                       |
| 194        | SF060000     | Middle box top                          |
| 195        | SF080000     | Left middle box side                    |
| 196        | SF100000     | Center box bar horizontal               |
| 197        | SF050000     | Box intersection                        |
| 198        | LA190000     | A tilde small                           |
| 199        | LA200000     | A tilde capital                         |
| 200        | SF380000     | Lower left box corner double            |
| 201        | SF390000     | Upper left box corner double            |
| 202        | SF400000     | Middle box bottom double                |
| 203        | SF410000     | Middle box top double                   |
| 204        | SF420000     | Left box side double                    |
| 205        | SF430000     | Center box bar horizontal double        |
| 206        | SF440000     | Box intersection double                 |
| 207        | SC010000     | International currency symbol           |
| 208        | LD630000     | eth Icelandic small                     |
| 209        | LD620000     | D stroke capital, Eth Icelandic capital |
| 210        | LE160000     | E circumflex capital                    |
| 211        | LE180000     | E diaeresis capital                     |
| 212        | LE140000     | E grave capital                         |
| 213        | LI610000     | I dotless small                         |
| 214        | LI120000     | I acute capital                         |
| 215        | LI160000     | I circumflex capital                    |
| 216        | LI180000     | I diaeresis capital                     |
| 217        | SF040000     | Lower right box corner                  |
| 218        | SF010000     | Upper left box corner                   |
| 219        | SF610000     | Solid fill character                    |
| 220        | SF570000     | Solid fill character, bottom half       |
| 221        | SM650000     | Vertical line, broken                   |
| 222        | LI140000     | I grave capital                         |
| 223        | SF600000     | Solid fill character, top half          |
| 224        | LO120000     | O acute capital                         |
| 225        | LS610000     | Sharp s small                           |
| 226        | LO160000     | O circumflex capital                    |
| 227        | LO140000     | O grave capital                         |
| 228        | LO190000     | O tilde small                           |
| 229        | LO200000     | O tilde capital                         |
| 230        | SM170000     | Micro symbol                            |
| 231        | LT630000     | Thorn Icelandic small                   |
| 232        | LT640000     | Thorn Icelandic capital                 |
| 233        | LU120000     | U acute capital                         |
| 234        | LU160000     | U circumflex capital                    |
| 235        | LU140000     | U grave capital                         |
| 236        | LY110000     | y acute small                           |
| 237        | LY120000     | Y acute capital                         |
| 238        | SM150000     | Overline                                |
| 239        | SD110000     | Acute accent                            |
| 240        | SP320000     | Syllable hyphen                         |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>                       |
|------------|--------------|--|
| 241        | SA020000     | Plus or minus sign                       |
| 242        | SM100000     | Double underscore                        |
| 243        | NF050000     | Three-quarters                           |
| 244        | SM250000     | Paragraph symbol (USA)                   |
| 245        | SM240000     | Section symbol (USA), paragraph (Europe) |
| 246        | SA060000     | Divide sign                              |
| 247        | SD410000     | Cedilla (or sedila) accent               |
| 248        | SM190000     | Degree symbol                            |
| 249        | SD170000     | Diaeresis, umlaut accent                 |
| 250        | SD630000     | Middle dot                               |
| 251        | ND011000     | One superscript                          |
| 252        | ND031000     | Three superscript                        |
| 253        | ND021000     | Two superscript                          |
| 254        | SM470000     | Solid square, histogram, square bullet   |
| 255        | SP300000     | Required space                           |
| 256        | SC060000     | Peseta sign                              |
| 257        | SM680000     | Start of line symbol                     |
| 258        | SF190000     | Right box side double to single          |
| 259        | SF200000     | Right box side single to double          |
| 260        | SF210000     | Upper right box corner single to double  |
| 261        | SF220000     | Upper right box corner double to single  |
| 262        | SF270000     | Lower right box corner single to double  |
| 263        | SF280000     | Lower right box corner double to single  |
| 264        | SF360000     | Left box side single to double           |
| 265        | SF370000     | Left box side double to single           |
| 266        | SF450000     | Middle box bottom single to double       |
| 267        | SF460000     | Middle box bottom double to single       |
| 268        | SF470000     | Middle box top double to single          |
| 269        | SF480000     | Middle box top single to double          |
| 270        | SF490000     | Lower left box corner double to single   |
| 271        | SF500000     | Lower left box corner single to double   |
| 272        | SF510000     | Upper left box corner single to double   |
| 273        | SF520000     | Upper left box corner double to single   |
| 274        | SF530000     | Box intersection single to double        |
| 275        | SF540000     | Box intersection double to single        |
| 276        | SF580000     | Solid fill character, left half          |
| 277        | SF590000     | Solid fill character, right half         |
| 278        | GA010000     | Alpha small                              |
| 279        | GG020000     | Gamma capital                            |
| 280        | GP010000     | Pi small                                 |
| 281        | GS020000     | Sigma capital                            |
| 282        | GS010000     | Sigma small                              |
| 283        | GT010000     | Tau small                                |
| 284        | GF020000     | Phi capital                              |
| 285        | GT620000     | Theta capital                            |
| 286        | GO320000     | Omega capital                            |
| 287        | GD010000     | Delta small                              |
| 288        | SA450000     | Infinity symbol                          |
| 289        | GF010000     | Phi small                                |
| 290        | GE010000     | Epsilon small                            |
| 291        | SA380000     | Intersection, logical product            |
| 292        | SA480000     | Indentity symbol, almost equal           |
| 293        | SA530000     | Greater than or equal sign               |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>                      |
|------------|--------------|---|
| 294        | SA520000     | Less than or equal sign                 |
| 295        | SS260000     | Upper integral symbol section           |
| 296        | SS270000     | Lower integral symbol section           |
| 297        | SA700000     | Nearly equals symbol                    |
| 298        | SA790000     | Product dot                             |
| 299        | SA800000     | Radical symbol                          |
| 300        | LN011000     | N small superscript                     |
| 301        | SD310000     | Macron accent                           |
| 302        | SD230000     | Breve accent                            |
| 303        | SD290000     | Overdot accent (over small Alpha)       |
| 304        | SD270000     | Overcircle accent                       |
| 305        | SD250000     | Double acute accent                     |
| 306        | SD430000     | Ogonek accent                           |
| 307        | SD210000     | Caron accent                            |
| 308        | SP190000     | Left single quote                       |
| 309        | SP200000     | Right single quote                      |
| 310        | SP210000     | Left double quotes                      |
| 311        | SP220000     | Right double quotes                     |
| 312        | SS680000     | Endash                                  |
| 313        | SM900000     | Emdash                                  |
| 314        | SD150000     | Circumflex accent                       |
| 315        | SD190000     | Tilde accent                            |
| 316        | SP260000     | Single quote on baseline (German lower) |
| 317        | SP230000     | Left lower double quotes                |
| 318        | SV520000     | Ellipsis                                |
| 319        | SM340000     | Dagger footnote indicator               |
| 320        | SM350000     | Double dagger footnote indicator        |
| 321        | SD150100     | Circumflex accent (over small alpha)    |
| 322        | SM560000     | Per mille symbol                        |
| 323        | LS220000     | S caron capital                         |
| 324        | SP270000     | French single open quote                |
| 325        | LO520000     | OE ligature capital                     |
| 326        | SD190100     | Tilde accent (over small alpha)         |
| 327        | SM540000     | Trademark symbol                        |
| 328        | LS210000     | s caron small                           |
| 329        | SP280000     | French single close quote               |
| 330        | LO510000     | oe ligature small                       |
| 331        | LY180000     | Y diaeresis capital                     |
| 332        | LG230000     | g Breve Small                           |
| 333        | LG240000     | G Breve Capital                         |
| 334        | LI300000     | I Overdot Capital                       |
| 335        | LS410000     | s Cedilla Small                         |
| 336        | LS420000     | S Cedilla Capital                       |
| 337        | LA230000     | a Breve Small                           |
| 338        | LA240000     | A Breve Capital                         |
| 339        | LA430000     | a Ogonek Small                          |
| 340        | LA440000     | A Ogonek Capital                        |
| 341        | LC110000     | c Acute Small                           |
| 342        | LC120000     | C Acute Capital                         |
| 343        | LC210000     | c Caron Small                           |
| 344        | LC220000     | C Caron Capital                         |
| 345        | LD210000     | d Caron Small                           |
| 346        | LD220000     | D Caron Capital                         |

| <b>UGL</b> | <b>GCGID</b> | <b>Description</b>     |
|------------|--------------|------------------------|
| 347        | LD610000     | d Stroke Small         |
| 348        | LE210000     | e Caron Small          |
| 349        | LE220000     | E Caron Capital        |
| 350        | LE430000     | e Ogenek Small         |
| 351        | LE440000     | E Ogenek Capital       |
| 352        | LL110000     | l Acute Small          |
| 353        | LL120000     | L Acute Capital        |
| 354        | LL210000     | l Caron Small          |
| 355        | LL220000     | L Caron Capital        |
| 356        | LL610000     | l Stroke Small         |
| 357        | LL620000     | L Stroke Capital       |
| 358        | LN110000     | n Acute Small          |
| 359        | LN120000     | N Acute Capital        |
| 360        | LN210000     | n Caron Small          |
| 361        | LN220000     | N Caron Capital        |
| 362        | LO250000     | o Double Acute Small   |
| 363        | LO260000     | O Double Acute Capital |
| 364        | LR110000     | r Acute Small          |
| 365        | LR120000     | R Acute Capital        |
| 366        | LR210000     | r Caron Small          |
| 367        | LR220000     | R Caron Capital        |
| 368        | LS110000     | s Acute Small          |
| 369        | LS120000     | S Acute Capital        |
| 370        | LT210000     | t Caron Small          |
| 371        | LT220000     | T Caron Capital        |
| 372        | LT410000     | t Cedilla Small        |
| 373        | LT420000     | T Cedilla Capital      |
| 374        | LU250000     | u Double Acute Small   |
| 375        | LU260000     | U Double Acute Capital |
| 376        | LU270000     | u Overcircle Small     |
| 377        | LU280000     | u Overcircle Capital   |
| 378        | LZ110000     | z Acute Small          |
| 379        | LZ120000     | Z Acute Capital        |
| 380        | LZ210000     | z Caron Small          |
| 381        | LZ220000     | Z Caron Capital        |
| 382        | LZ290000     | z Overdot Small        |
| 383        | LZ300000     | Z Overdot Capital      |



# ASCII Code Pages

| 1  |                                  | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160          | 176 | 192 | 208 | 224 | 240          |
|----|----------------------------------|----|----|----|----|----|----|----|-----|-----|-----|--------------|-----|-----|-----|-----|--------------|
|    | 2 <sup>A</sup><br>↓ <sub>B</sub> | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-           | B-  | C-  | D-  | E-  | F-           |
| 0  | -0                               |    | ▶  |    | 0  | @  | P  | χ  | p   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | a   | X            |
| 1  | -1                               | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | b   | #            |
| 2  | -2                               | ☻  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | Q   | /            |
| 3  | -3                               | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?            |     | ⊥   | ⊥   | f   | &            |
| 4  | -4                               | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | S   | ƒ            |
| 5  | -5                               | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | S   | J            |
| 6  | -6                               | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | <sup>a</sup> | ⊥   | ⊥   | ⊥   | m   | :            |
| 7  | -7                               | •  | ↕  | '  | 7  | G  | W  | g  | w   | ?   | ?   | <sup>o</sup> | ⊥   | ⊥   | ⊥   | t   | Z            |
| 8  | -8                               | ◼  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | D   | °            |
| 9  | -9                               | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | R   | •            |
| 10 | -A                               | ◼  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | L   | •            |
| 11 | -B                               | ♂  | ←  | +  | ;  | K  | [  | k  | {   | ?   | ?   |              | ⊥   | ⊥   | ◼   | W   | !            |
| 12 | -C                               | ♀  | ⊥  | ,  | <  | L  | \  | l  |     | ?   | ?   |              | ⊥   | ⊥   | ◼   | B   | <sup>n</sup> |
| 13 | -D                               | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | ?   | ¥   | ?            | ⊥   | ⊥   | ◼   | ∅   | <sup>2</sup> |
| 14 | -E                               | ♪  | ▲  | .  | >  | N  | ^  | n  | χ   | ?   | Pt  | ?            | ⊥   | ⊥   | ◼   | e   | ◼            |
| 15 | -F                               | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | â   | ?            | ⊥   | ⊥   | ◼   | ∅   |              |

Figure 32-2. US-English: ASCII Code Page 437

| 1  |                                  | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|----|----------------------------------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|    | 2 <sub>A</sub><br>↓ <sub>B</sub> | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-  | B-  | C-  | D-  | E-  | F-  |
| 0  | -0                               |    | ▶  |    | 0  | @  | P  | λ  | p   | ?   | ?   | ?   | ⋮   | └   | >   | ?   | -   |
| 1  | -1                               | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | ?   | ⋮   | └   | _   | b   | #   |
| 2  | -2                               | ☹  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?   | ⋮   | └   | ?   | ?   | =   |
| 3  | -3                               | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?   |     | └   | ?   | ?   |     |
| 4  | -4                               | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | ?   | └   | —   | ?   | ?   | ?   |
| 5  | -5                               | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | ?   | ?   | ?   | +   | 1   | ?   | ?   |
| 6  | -6                               | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | ª   | ?   | ?   | ?   | ?   | :   |
| 7  | -7                               | •  | ↕  | '  | 7  | G  | W  | g  | w   | ?   | ?   | º   | ?   | ?   | ?   | ?   | ˜   |
| 8  | -8                               | ◼  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ?   | ?   | +   | └   | ?   | ˆ   | ◦   |
| 9  | -9                               | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ;   | ≡   | ≡   | └   | ?   | ˆ   |
| 10 | -A                               | ◉  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ∟   |     | ≡   | └   | ?   | •   |
| 11 | -B                               | ♂  | ←  | +  | ;  | K  | I  | k  | {   | ?   | ?   |     | ≡   | ≡   | ◼   | ?   | 1   |
| 12 | -C                               | ♀  | └  | ,  | <  | L  | \  | l  |     | ?   | ?   |     | ≡   | ≡   | ◼   | ˆ   | 3   |
| 13 | -D                               | ♪  | ↔  | -  | =  | M  | J  | m  | }   | ?   | ?   | ?   | ?   | ≡   |     | ˆ   | 2   |
| 14 | -E                               | ♫  | ▲  | .  | >  | N  | ^  | n  | ×   | ?   | x   | ?   | ≡   | ≡   | ?   | -   | ◼   |
| 15 | -F                               | ☼  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | å   | ?   | └   | ⊗   | ◼   | '   |     |

Figure 32-3. Latin 1 Multilingual: ASCII Code Page 850

| 1  |                                  | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|----|----------------------------------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|    | 2 <sup>A</sup><br>↓ <sub>B</sub> | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-  | B-  | C-  | D-  | E-  | F-  |
| 0  | -0                               |    | ▶  |    | 0  | @  | P  | λ  | p   | ?   | ?   | ?   | ⋮   | ┌   | d   | ?   | ?   |
| 1  | -1                               | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | Ł   | ?   | ⋮   | └   | —   | b   | "   |
| 2  | -2                               | ☹  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | Í   | ?   | ⋮   | └   | Ď   | ?   | ç   |
| 3  | -3                               | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?   |     | └   | ?   | Ń   | ˘   |
| 4  | -4                               | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | Ą   | └   | —   | ď   | ń   | ˘   |
| 5  | -5                               | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ű   | Ł   | ą   | ?   | +   | Ń   | ń   | ?   |
| 6  | -6                               | ♠  | —  | &  | 6  | F  | V  | f  | v   | ć   | Ĭ   | Ż   | ?   | Ā   | ?   | Š   | :   |
| 7  | -7                               | •  | ↕  | ,  | 7  | G  | W  | g  | w   | ?   | Ś   | ż   | Ě   | ā   | ?   | š   | =   |
| 8  | -8                               | ■  | ↑  | (  | 8  | H  | X  | h  | x   | ł   | ś   | Ę   | Ş   | ┌   | ě   | Ř   | °   |
| 9  | -9                               | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ę   | ≡   | ┌   | └   | ?   | ˆ   |
| 10 | -A                               | ◉  | →  | *  | :  | J  | Z  | j  | z   | Ō   | ?   | ⌋   |     | └   | ┌   | ř   | •   |
| 11 | -B                               | ♂  | ←  | +  | ;  | K  | l  | k  | {   | õ   | Ŧ   | ž   | ⌋   | └   | ■   | Ů   | ů   |
| 12 | -C                               | ♀  | └  | ,  | <  | L  | \  | l  | l   | ?   | ř   | č   | ⌋   | └   | ■   | ý   | ř   |
| 13 | -D                               | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | Ž   | L   | ş   | Ž   | =   | Ṭ   | Ý   | ř   |
| 14 | -E                               | ♫  | ▲  | .  | >  | N  | ^  | n  | ×   | ?   | x   | ?   | ž   | └   | Ů   | ṭ   | ■   |
| 15 | -F                               | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | Ć   | č   | ?   | ⌋   | ⊗   | ■   | ,   |     |

Figure 32-4. Latin 2 Multilingual: ASCII Code Page 852

| 1  | 0                                | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |    |
|----|----------------------------------|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|
|    | 2 <sub>A</sub><br>↓ <sub>B</sub> | 0- | 1- | 2- | 3- | 4- | 5- | 6-  | 7-  | 8-  | 9-  | A-  | B-  | C-  | D-  | E-  | F- |
| 0  | -0                               |    | ▶  |    | 0  | @  | P  | λ   | p   | ?   | ?   | ?   | ⋮   | └   | °   | ?   | ?  |
| 1  | -1                               | ☺  | ◀  | !  | 1  | A  | Q  | a   | q   | ?   | ?   | ?   | ⋮   | └   | ª   | b   | ±  |
| 2  | -2                               | ☹  | ↕  | "  | 2  | B  | R  | b   | r   | ?   | ?   | ?   | ⋮   | └   | ?   | ?   |    |
| 3  | -3                               | ♥  | !! | #  | 3  | C  | S  | c   | s   | ?   | ?   | ?   |     | └   | ?   | ?   |    |
| 4  | -4                               | ♦  | ?  | \$ | 4  | D  | T  | d   | t   | ?   | ?   | ?   | └   | —   | ?   | ?   | ?  |
| 5  | -5                               | ♣  | ?  | %  | 5  | E  | U  | e   | u   | ?   | ?   | ?   | ?   | +   |     | ?   | ?  |
| 6  | -6                               | ♠  | —  | &  | 6  | F  | V  | f   | v   | ?   | ?   | Ğ   | ?   | ?   | ?   | m   | :  |
| 7  | -7                               | •  | ↕  | ,  | 7  | G  | W  | g   | w   | ?   | ?   | ğ   | ?   | ?   | ?   |     | =  |
| 8  | -8                               | ■  | ↑  | (  | 8  | H  | X  | h   | x   | ?   | İ   | ?   | +   | └   | ?   | x   | °  |
| 9  | -9                               | ○  | ↓  | )  | 9  | I  | Y  | i   | y   | ?   | ?   | ;   | └   | └   | └   | ?   | .. |
| 10 | -A                               | ●  | →  | *  | :  | J  | Z  | j   | z   | ?   | ?   | └   |     | └   | └   | ?   | •  |
| 11 | -B                               | ♂  | ←  | +  | ;  | K  | [  | k   | {   | ?   | ?   |     | └   | └   | ■   | ?   | 1  |
| 12 | -C                               | ♀  | └  | ,  | <  | L  | \  | l   |     | ?   | ?   |     | └   | └   | ■   | ?   | 3  |
| 13 | -D                               | ♪  | ↔  | -  | =  | M  | ]  | m   | }   | ı   | ?   | ?   | ?   | ==  | ı   | ?   | 2  |
| 14 | -E                               | ♫  | ▲  | .  | >  | N  | ^  | n   | ×   | ?   | §   | ?   | ¥   | └   | ?   | ?   | ■  |
| 15 | -F                               | ☀  | ▼  | /  | ?  | O  | _  | o   | △   | ?   | §   | ?   | └   | ⊗   | ■   | ,   |    |

Figure 32-5. Turkey: ASCII Code Page 857

| 1  |                  | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|----|------------------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|    | 2<br>A<br>↓<br>B | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-  | B-  | C-  | D-  | E-  | F-  |
| 0  | -0               |    | ▶  |    | 0  | @  | P  | λ  | p   | ?   | ?   | ?   | ⋮   | L   | ⊥   | a   | X   |
| 1  | -1               | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | ?   | ⋮   | ⊥   | ⊥   | b   | #   |
| 2  | -2               | ☺  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?   | ⋮   | ⊥   | ⊥   | Q   | /   |
| 3  | -3               | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?   |     | ⊥   | ⊥   | f   | &   |
| 4  | -4               | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | ?   | ⊥   | ⊥   | ⊥   | S   | f   |
| 5  | -5               | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | ?   | ?   | ⊥   | ⊥   | ⊥   | S   | J   |
| 6  | -6               | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | ª   | ⊥   | ⊥   | ⊥   | m   | :   |
| 7  | -7               | •  | ↕  | ,  | 7  | G  | W  | g  | w   | ?   | ?   | º   | ⊥   | ⊥   | ⊥   | t   | Z   |
| 8  | -8               | ■  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ?   | ?   | ⊥   | ⊥   | ⊥   | D   | °   |
| 9  | -9               | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ?   | ⊥   | ⊥   | ⊥   | R   | •   |
| 10 | -A               | ⊙  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | L   | •   |
| 11 | -B               | ♂  | ←  | +  | ;  | K  | [  | k  | {   | ?   | ?   |     | ⊥   | ⊥   | ■   | w   | !   |
| 12 | -C               | ♀  | ⊥  | ,  | <  | L  | \  | l  |     | ?   | ?   |     | ⊥   | ⊥   | ■   | B   | n   |
| 13 | -D               | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | ?   | ?   | ?   | ⊥   | ⊥   | ■   | ∅   | ²   |
| 14 | -E               | ♪  | ▲  | .  | >  | N  | ^  | n  | ×   | ?   | Pts | ?   | ⊥   | ⊥   | ■   | e   | ■   |
| 15 | -F               | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | ?   | ?   | ⊥   | ⊥   | ■   | ∅   |     |

Figure 32-6. Portuguese: ASCII Code Page 860

| 1  |                          | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|----|--------------------------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|    | 2 <sub>↓</sub><br>A<br>B | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-  | B-  | C-  | D-  | E-  | F-  |
| 0  | -0                       | ▶  |    | 0  | @  | P  | λ  | p  | ?   | ?   | ?   | ⋮   | L   | ⊥   | a   | ×   |     |
| 1  | -1                       | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | ?   | ⋮   | ⊥   | ⊥   | b   | #   |
| 2  | -2                       | ☹  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?   | ⋮   | ⊥   | ⊥   | Q   | /   |
| 3  | -3                       | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?   |     | ⊥   | ⊥   | f   | &   |
| 4  | -4                       | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | Á   | ⊥   | ⊥   | ⊥   | ⊥   | ƒ   |
| 5  | -5                       | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | Ö   | Í   | ⊥   | ⊥   | ⊥   | ⊥   | J   |
| 6  | -6                       | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | :   |
| 7  | -7                       | •  | ↕  | '  | 7  | G  | W  | g  | w   | ?   | Ý   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | Z   |
| 8  | -8                       | ◼  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ý   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ◦   |
| 9  | -9                       | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ⊥   | •   |
| 10 | -A                       | ◉  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ⊥   | •   |
| 11 | -B                       | ♂  | ←  | +  | ;  | K  | [  | k  | {   | —   | ?   |     | ⊥   | ⊥   | ⊥   | ⊥   | !   |
| 12 | -C                       | ♀  | ⊥  | ,  | <  | L  | \  | l  |     | >   | ?   |     | ⊥   | ⊥   | ⊥   | ⊥   | n   |
| 13 | -D                       | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | Ó   | ?   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ²   |
| 14 | -E                       | 🎵  | ▲  | .  | >  | N  | ^  | n  | ×   | ?   | Pts | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ■   |
| 15 | -F                       | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | å   | ?   | ⊥   | ⊥   | ⊥   | ⊥   | ∅   |

Figure 32-7. Iceland: ASCII Code Page 861

| 1  |                                  | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160          | 176 | 192 | 208 | 224 | 240 |
|----|----------------------------------|----|----|----|----|----|----|----|-----|-----|-----|--------------|-----|-----|-----|-----|-----|
|    | 2 <sub>A</sub><br>↓ <sub>B</sub> | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-           | B-  | C-  | D-  | E-  | F-  |
| 0  | -0                               |    | ▶  |    | 0  | @  | P  | λ  | p   | ?   | ?   | !            | ⋮   | L   | ⊥   | a   | X   |
| 1  | -1                               | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | '            | ⋮   | ⊥   | ⊥   | b   | #   |
| 2  | -2                               | ☹  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | Q   | /   |
| 3  | -3                               | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?            |     | ⊥   | ⊥   | f   | &   |
| 4  | -4                               | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | >            | ⊥   | ⊥   | ⊥   | S   | ƒ   |
| 5  | -5                               | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | ?   | =            | ⊥   | ⊥   | ⊥   | S   | J   |
| 6  | -6                               | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | <sup>3</sup> | ⊥   | ⊥   | ⊥   | m   | :   |
| 7  | -7                               | •  | ↕  | '  | 7  | G  | W  | g  | w   | ?   | ?   | -            | ⊥   | ⊥   | ⊥   | t   | Z   |
| 8  | -8                               | ■  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ∅   | ?            | ⊥   | ⊥   | ⊥   | D   | °   |
| 9  | -9                               | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | R   | •   |
| 10 | -A                               | ◼  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | L   | •   |
| 11 | -B                               | ♂  | ←  | +  | ;  | K  | [  | k  | {   | ?   | ?   |              | ⊥   | ⊥   | ⊥   | w   | !   |
| 12 | -C                               | ♀  | ⊥  | ,  | <  | L  | \  | l  |     | ?   | ?   |              | ⊥   | ⊥   | ⊥   | B   | n   |
| 13 | -D                               | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | =   | ?   |              | ⊥   | ⊥   | ⊥   | ∅   | 2   |
| 14 | -E                               | ♫  | ▲  | .  | >  | N  | ^  | n  | X   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | e   | ■   |
| 15 | -F                               | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | å   | ?            | ⊥   | ⊥   | ⊥   | ∅   |     |

Figure 32-8. Canadian-French: ASCII Code Page 863

| 1  |   | 0  | 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160          | 176 | 192 | 208 | 224 | 240 |
|----|---|----|----|----|----|----|----|----|-----|-----|-----|--------------|-----|-----|-----|-----|-----|
|    | 2 <sub>↓</sub> <sup>A</sup> <sub>↓</sub> <sup>B</sup> | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7-  | 8-  | 9-  | A-           | B-  | C-  | D-  | E-  | F-  |
| 0  | -0  | ▶  |    | 0  | @  | P  | λ  | p  | ?   | ?   | ?   | ⋮            | ⊥   | ⊥   | a   | ×   |     |
| 1  | -1  | ☺  | ◀  | !  | 1  | A  | Q  | a  | q   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | b   | #   |
| 2  | -2  | ☹  | ↕  | "  | 2  | B  | R  | b  | r   | ?   | ?   | ?            | ⋮   | ⊥   | ⊥   | Q   | /   |
| 3  | -3  | ♥  | !! | #  | 3  | C  | S  | c  | s   | ?   | ?   | ?            |     | ⊥   | ⊥   | f   | &   |
| 4  | -4  | ♦  | ?  | \$ | 4  | D  | T  | d  | t   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | S   | ƒ   |
| 5  | -5  | ♣  | ?  | %  | 5  | E  | U  | e  | u   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | S   | J   |
| 6  | -6  | ♠  | —  | &  | 6  | F  | V  | f  | v   | ?   | ?   | <sup>a</sup> | ⊥   | ⊥   | ⊥   | m   | :   |
| 7  | -7  | •  | ↕  | '  | 7  | G  | W  | g  | w   | ?   | ?   | <sup>o</sup> | ⊥   | ⊥   | ⊥   | t   | Z   |
| 8  | -8  | ■  | ↑  | (  | 8  | H  | X  | h  | x   | ?   | ?   | ?            | ⊥   | ⊥   | ⊥   | D   | °   |
| 9  | -9  | ○  | ↓  | )  | 9  | I  | Y  | i  | y   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | R   | •   |
| 10 | -A  | ●  | →  | *  | :  | J  | Z  | j  | z   | ?   | ?   | ⊥            | ⊥   | ⊥   | ⊥   | L   | •   |
| 11 | -B  | ♂  | ←  | +  | ;  | K  | [  | k  | {   | ?   | ?   |              | ⊥   | ⊥   | ■   | w   | !   |
| 12 | -C  | ♀  | ⊥  | ,  | <  | L  | \  | l  |     | ?   | ?   |              | ⊥   | ⊥   | ■   | B   | n   |
| 13 | -D  | ♪  | ↔  | -  | =  | M  | ]  | m  | }   | ?   | ?   | ?            | ⊥   | ⊥   | ■   | ∅   | 2   |
| 14 | -E  | ♫  | ▲  | .  | >  | N  | ^  | n  | ×   | ?   | Pt  | ?            | ⊥   | ⊥   | ■   | e   | ■   |
| 15 | -F  | ☀  | ▼  | /  | ?  | O  | _  | o  | △   | ?   | å   | œ            | ⊥   | ⊥   | ■   | ∅   |     |

Figure 32-9. Norwegian: ASCII Code Page 865



| →<br>↓    | 0- | 1- | 2-     | 3- | 4- | 5- | 6- | 7- | 8-  | 9- | A-         | B- | C- | D- | E- | F- |
|-----------|----|----|--------|----|----|----|----|----|-----|----|------------|----|----|----|----|----|
| <b>-0</b> |    |    | (SP) 0 | @  | P  | `  | p  |    |     |    | (RSP) °    | ?  | Ɖ  | ?  | ø  |    |
| <b>-1</b> |    |    | !      | 1  | A  | Q  | a  | q  |     | ‘  | ?          | ƒ  | ?  | ?  | ?  | ?  |
| <b>-2</b> |    |    | "      | 2  | B  | R  | b  | r  | ,   | ’  | ?          | ²  | ?  | ?  | ?  | ?  |
| <b>-3</b> |    |    | #      | 3  | C  | S  | c  | s  |     | ?  | ?          | ³  | ?  | ?  | ?  | ?  |
| <b>-4</b> | -  |    | \$     | 4  | D  | T  | d  | t  | ?   | ?  | ⊗          | ’  | ?  | ?  | ?  | ?  |
| <b>-5</b> | ˘  |    | %      | 5  | E  | U  | e  | u  | ... | ●  | ¥          | μ  | ?  | ?  | ?  | ?  |
| <b>-6</b> | ·  |    | &      | 6  | F  | V  | f  | v  | †   | ?  | !          | ¶  | ?  | ?  | ?  | ?  |
| <b>-7</b> |    |    | ’      | 7  | G  | W  | g  | w  | ‡   | ?  | §          | •  | ?  | x  | ?  | ÷  |
| <b>-8</b> | °  |    | (      | 8  | H  | X  | h  | x  | ^   | ~  | ¨          | ˘  | ?  | ?  | ?  | ?  |
| <b>-9</b> |    |    | )      | 9  | I  | Y  | i  | y  | %   | ™  | ©          | 1  | ?  | ?  | ?  | ?  |
| <b>-A</b> | "  |    | *      | :  | J  | Z  | j  | z  | Š   | š  | ª          | º  | ?  | ?  | ?  | ?  |
| <b>-B</b> | ˙  |    | +      | ;  | K  | [  | k  | {  | <   | >  | ?          | ?  | ?  | ?  | ?  | ?  |
| <b>-C</b> | ˘  |    | ,      | <  | L  | \  | l  |    | ?   | ?  | ⌋          | ¼  | ?  | ?  | ?  | ?  |
| <b>-D</b> |    |    | -      | =  | M  | ]  | m  | }  |     |    | ¿<br>(SHY) | ½  | ?  | Ÿ  | ?  | ý  |
| <b>-E</b> |    |    | .      | >  | N  | ^  | n  | ~  |     |    | ®          | ¾  | ?  | Ɔ  | ?  | ƒ  |
| <b>-F</b> |    |    | /      | ?  | O  | _  | o  |    |     |    | ?          | ?  | ?  | ?  | ?  | ?  |

Figure 32-10. Desktop Publishing: ASCII Code Page 1004

# EBCDIC Code Pages

| ↓  | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0 |    |    |    |    |    | &  | -  | ø  | ø  | °  | μ  | ϕ  | {  | }  | \  | 0  |
| -1 |    |    |    |    |    | é  | /  | É  | a  | j  | —  | l  | A  | J  | ÷  | 1  |
| -2 |    |    |    |    | â  | ê  | Â  | Ê  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| -3 |    |    |    |    | ä  | ë  | Ä  | Ë  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| -4 |    |    |    |    | à  | è  | À  | È  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| -5 |    |    |    |    | á  | í  | Á  | Í  | e  | n  | v  | §  | E  | N  | V  | 5  |
| -6 |    |    |    |    | ã  | î  | Ã  | Î  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| -7 |    |    |    |    | å  | ï  | Å  | Ï  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| -8 |    |    |    |    | ç  | ì  | Ç  | Ì  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| -9 |    |    |    |    | ñ  | ß  | Ñ  | `  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| -A |    |    |    |    | \$ | !  | !  | :  | «  | ª  | ¡  | ^  | —  | 1  | 2  | 3  |
| -B |    |    |    |    | .  | £  | ,  | #  | »  | º  | ¿  | ]  | ô  | û  | Ô  | Û  |
| -C |    |    |    |    | <  | *  | %  | @  | ð  | æ  | Ð  | ~  | ö  | ü  | Ö  | Ü  |
| -D |    |    |    |    | (  | )  | _  | '  | ý  | ¸  | Ý  | ¨  | ò  | ù  | Ò  | Ù  |
| -E |    |    |    |    | +  | ;  | >  | =  | þ  | Æ  | Þ  | '  | ó  | ú  | Ó  | Ú  |
| -F |    |    |    |    |    | ∟  | ?  | "  | ±  | ∅  | ®  | √  | õ  | ÿ  | Õ  | EO |

Figure 32-11. US-English: EBCDIC Code Page 037

| →<br>↓    | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ?  | ?  | °            | μ  | ?  | ?  | ?  | ?  | 0  |
| <b>-1</b> |    |    |    |    |    | ?  | /  | ?  | a  | j            | ?  | ?  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | ?  | ?  | ?  | ?  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | {  | ?  | [  | ?  | c  | l            | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | ?  | ?  | ?  | ?  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | ?  | ?  | ?  | ?  | e  | n            | v  | @  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ?  | ?  | ?  | ?  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | ?  | ?  | ?  | ?  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | ?  | ?  | ?  | ?  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ?  | ~  | ?  | ∖  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | ?  | ?  | ?  | ∴  | ?  | <sup>a</sup> | ?  | ⌊  | ?  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | #  | ?  | °            | ?  |    | ?  | ?  | ?  | ?  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | §  | ö  | ?            | Ð  | —  | '  | }  | \  |    |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ý  | ˆ            | ÿ  | ˆ  | ?  | ?  | ?  | ?  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | ?            | Þ  | '  | ?  | ?  | ?  | ?  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ‡  | ⊗            | ®  | √  | ?  | ?  | ?  | EO |

Figure 32-12. Austrian/German: EBCDIC Code Page 273

| →<br>↓    | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ?  | ?  | °            | μ  | ?  | ?  | ?  | ?  | 0  |
| <b>-1</b> |    |    |    |    |    | {  | /  | ?  | a  | j            | ¨  | ?  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | ?  | ?  | ?  | ?  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ?  | ?  | ?  | ?  | c  | l            | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | @  | }  | ?  | ?  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | ?  | ?  | ?  | ?  | e  | n            | v  | §  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ?  | ?  | ?  | ?  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | ?  | ?  | ?  | ?  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | \  | ?  | ?  | ?  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ?  | ?  | ?  | \  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | [  | [  | ?  | :  | ?  | <sup>a</sup> | ?  | ⌋  | ?  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | #  | ?  | °            | ?  |    | ?  | ?  | ?  | ?  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | ?  | ø  | ?            | Ð  | —  | ?  | ?  | ?  | ?  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ý  | ˆ            | ÿ  | ~  | ?  | '  | ?  | ?  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | b  | ?            | Ɔ  | '  | ?  | ?  | ?  | ?  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ‡  | ⊗            | ®  | ∨  | ?  | ?  | ?  | EO |

Figure 32-13. Belgian: EBCDIC Code Page 274 (supported for migration purposes)

| ↓  | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|----|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| -0 |    |    |    |    | &  | -  | !  | @  | °  | μ            | ¢  | æ  | å  | \  | 0  |    |
| -1 |    |    |    |    | é  | /  | É  | a  | j  | ü            | £  | A  | J  | ÷  | 1  |    |
| -2 |    |    |    |    | â  | ê  | Â  | Ê  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| -3 |    |    |    |    | ä  | ë  | Ä  | Ë  | c  | l            | t  | ·  | C  | L  | T  | 3  |
| -4 |    |    |    |    | à  | è  | À  | È  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| -5 |    |    |    |    | á  | í  | Á  | Í  | e  | n            | v  | §  | E  | N  | V  | 5  |
| -6 |    |    |    |    | ã  | î  | Ã  | Î  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| -7 |    |    |    |    | {  | ï  | \$ | Ï  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| -8 |    |    |    |    | ç  | ì  | Ç  | Ì  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| -9 |    |    |    |    | ñ  | ß  | Ñ  | `  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| -A |    |    |    |    | #  | □  | ø  | :  | «  | <sup>a</sup> | ¡  | ¬  | —  | 1  | 2  | 3  |
| -B |    |    |    |    | .  | Å  | ,  | Æ  | »  | °            | ¿  |    | ô  | û  | Ô  | Û  |
| -C |    |    |    |    | <  | *  | %  | ø  | ö  | }            | Ð  | —  | ö  | ~  | Ö  | Ü  |
| -D |    |    |    |    | (  | )  | _  | '  | ý  | ˘            | Ý  | ¨  | ò  | ù  | Ò  | Ù  |
| -E |    |    |    |    | +  | ;  | >  | =  | þ  | [            | Þ  | '  | ó  | ú  | Ó  | Ú  |
| -F |    |    |    |    | !  | ^  | ?  | "  | ‡  | ]            | ®  | √  | õ  | ÿ  | Õ  | EO |

Figure 32-14. Danish/Norwegian: EBCDIC Code Page 277

| ↓         | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ?  | ?  | °  | μ  | ?  | ?  | ?  | ?  | 0  |
| <b>-1</b> |    |    |    |    |    | {  | /  | ?  | a  | j  | ¨  | ?  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | @  | }  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | §  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ?  | ?  | ?  | ?  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | \  | ?  | ?  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ?  | ?  | ?  | \  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | [  | [  | ?  | :  | ?  | ª  | ?  | ¬  | ?  | ¹  | ²  | ³  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | #  | ?  | º  | ?  | ı  | ?  | ?  | ?  | ?  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | ?  | ø  | ?  | Ð  | —  | ?  | ?  | ?  | ?  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ý  | ˆ  | Ÿ  | ~  | ?  | '  | ?  | ?  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | ?  | Ɔ  | '  | ?  | ?  | ?  | ?  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ‡  | ⊗  | ®  | √  | ?  | ?  | ?  | EO |

Figure 32-15. Finnish/Swedish: EBCDIC Code Page 278

| →<br>↓    | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ?  | ?  | [  | μ  | ?  | ?  | ?  | ?  | 0  |
| <b>-1</b> |    |    |    |    |    | ]  | /  | ?  | a  | j  | ?  | #  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | {  | }  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | @  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ?  | ?  | ?  | ?  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | \  | ~  | ?  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ?  | ?  | ?  | ?  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | °  | ?  | ?  | :  | ?  | ª  | ?  | ¬  | ?  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | ?  | ?  | º  | ?  |    | ?  | ?  | ?  | ?  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | §  | ø  | ?  | Ð  | —  | ?  | ?  | ?  | ?  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ÿ  | ˆ  | Ÿ  | ¨  | '  | \  | ?  | ?  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | ?  | Ɔ  | '  | ?  | ?  | ?  | ?  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ‡  | ⊗  | ®  | √  | ?  | ?  | ?  | EO |

Figure 32-16. Italian: EBCDIC Code Page 280

| ↓  | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0 |    |    |    |    | &  | -  | ?  | ?  | °  | μ  | ?  | ?  | ·  | ?  | ?  | 0  |
| -1 |    |    |    |    | ?  | /  | ?  | a  | j  | ?  | ?  | A  | J  | ÷  | ?  | 1  |
| -2 |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | ?  | 2  |
| -3 |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | ?  | 3  |
| -4 |    |    |    | ?  | ?  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | ?  | 4  |
| -5 |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | §  | E  | N  | V  | ?  | 5  |
| -6 |    |    |    | {  | ?  | #  | ?  | f  | o  | w  | ¶  | F  | O  | W  | ?  | 6  |
| -7 |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | ?  | 7  |
| -8 |    |    |    | ~  | ?  | \  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | ?  | 8  |
| -9 |    |    |    | ?  | ?  | ?  | ?  | i  | r  | z  | ¾  | I  | R  | Z  | ?  | 9  |
| -A |    |    |    | [  | ]  | ?  | :  | ?  | ª  | ?  | ¬  | ?  | ¹  | ²  | ³  |    |
| -B |    |    |    | .  | \$ | ,  | ?  | ?  | º  | ?  | ¡  | ?  | ?  | ?  | ?  |    |
| -C |    |    |    | <  | *  | %  | ?  | ø  | ?  | Ð  | —  | ?  | ?  | ?  | ?  |    |
| -D |    |    |    | (  | )  | _  | '  | ÿ  | ¸  | Ý  | ¨  | ?  | '  | ?  | ?  |    |
| -E |    |    |    | +  | ;  | >  | =  | þ  | ?  | Þ  | }  | ?  | ?  | ?  | ?  |    |
| -F |    |    |    | !  | ^  | ?  | "  | ‡  | ∅  | ®  | √  | '  | ?  | @  | EO |    |

Figure 32-17. Portuguese: EBCDIC Code Page 282 (supported for migration purposes)



| ↓  | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0 |    |    |    |    |    | &  | -  | ?  | ?  | °  | μ  | ?  | {  | }  | \  | 0  |
| -1 |    |    |    |    |    | ?  | /  | ?  | a  | j  | ¨  | ?  | A  | J  | ÷  | 1  |
| -2 |    |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| -3 |    |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| -4 |    |    |    |    | ?  | ?  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| -5 |    |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | §  | E  | N  | V  | 5  |
| -6 |    |    |    |    | ?  | ?  | ?  | ?  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| -7 |    |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| -8 |    |    |    |    | ?  | ?  | ?  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| -9 |    |    |    |    | '  | ?  | #  | `  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| -A |    |    |    |    | [  | ]  | ?  | :  | ?  | ª  | ?  | ^  | ?  | ¹  | ²  | ³  |
| -B |    |    |    |    | .  | \$ | ,  | ?  | ?  | º  | ?  | !  | ?  | ?  | ?  | ?  |
| -C |    |    |    |    | <  | *  | %  | @  | ø  | ?  | Ð  | —  | ?  | ?  | ?  | ?  |
| -D |    |    |    |    | (  | )  | _  | '  | ÿ  | ¸  | Ý  | ~  | ?  | ?  | ?  | ?  |
| -E |    |    |    |    | +  | ;  | >  | =  | þ  | ?  | Þ  | '  | ?  | ?  | ?  | ?  |
| -F |    |    |    |    |    | ¬  | ?  | "  | ‡  | œ  | ®  | √  | ?  | ?  | ?  | EO |

Figure 32-18. Spanish: EBCDIC Code Page 284

| →<br>↓    | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ø  | ø  | °            | μ  | ¢  | {  | }  | \  | 0  |
| <b>-1</b> |    |    |    |    |    | é  | /  | É  | a  | j            | —  | l  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | â  | ê  | Â  | Ê  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ä  | ë  | Ä  | Ë  | c  | l            | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | à  | è  | À  | È  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | á  | í  | Á  | Í  | e  | n            | v  | §  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ã  | î  | Ã  | Î  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | å  | ï  | Å  | Ï  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | ç  | ì  | Ç  | Ì  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ñ  | ß  | Ñ  | `  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | \$ | !  | !  | :  | «  | <sup>a</sup> | i  | ^  | -  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | £  | ,  | #  | »  | <sup>o</sup> | ı  | l  | ô  | û  | Ô  | Û  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | @  | ö  | æ            | Ð  | ~  | ö  | ü  | Ö  | Ü  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ý  | ˆ            | Ý  | ¨  | ò  | ù  | Ò  | Ù  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | Æ            | Ɔ  | '  | ó  | ú  | Ó  | Ú  |
| <b>-F</b> |    |    |    |    |    | ⌋  | ?  | "  | ±  | ∅            | ®  | √  | õ  | ÿ  | Õ  | EO |

Figure 32-19. UK-English: EBCDIC Code Page 285

| →<br>↓    | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ?  | ?  | [  | `  | ?  | ?  | ?  | ?  | 0  |
| <b>-1</b> |    |    |    |    |    | {  | /  | ?  | a  | j  | .. | #  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | @  | }  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | ]  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ?  | ?  | ?  | ?  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | \  | ?  | ?  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ?  | ?  | ?  | μ  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | °  | §  | ?  | :  | ?  | ª  | ?  | ¬  | ?  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | ?  | ?  | º  | ?  | l  | ?  | ?  | ?  | ?  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | ?  | ø  | ?  | Ð  | —  | ?  | ?  | ?  | ?  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ý  | _  | Ý  | ~  | ?  | '  | ?  | ?  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | ?  | Þ  | '  | ?  | ?  | ?  | ?  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ‡  | ⊗  | ®  | √  | ?  | ?  | ?  | EO |

Figure 32-20. French: EBCDIC Code Page 297

| ↓         | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|-----------|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| <b>-0</b> |    |    |    |    |    | &  | -  | ø  | @  | °            | μ  | ¢  | {  | }  | \  | 0  |
| <b>-1</b> |    |    |    |    |    | é  | /  | É  | a  | j            | ~  | £  | A  | J  | ÷  | 1  |
| <b>-2</b> |    |    |    |    | â  | ê  | Â  | Ê  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| <b>-3</b> |    |    |    |    | ä  | ë  | Ä  | Ë  | c  | l            | t  | •  | C  | L  | T  | 3  |
| <b>-4</b> |    |    |    |    | à  | è  | À  | È  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| <b>-5</b> |    |    |    |    | á  | í  | Á  | Í  | e  | n            | v  | §  | E  | N  | V  | 5  |
| <b>-6</b> |    |    |    |    | ã  | î  | Ã  | Î  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| <b>-7</b> |    |    |    |    | å  | ï  | Å  | Ï  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| <b>-8</b> |    |    |    |    | ç  | ì  | Ç  | Ì  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| <b>-9</b> |    |    |    |    | ñ  | ß  | Ñ  | `  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| <b>-A</b> |    |    |    |    | [  | ]  | !  | :  | «  | <sup>a</sup> | ¡  | ¬  | —  | 1  | 2  | 3  |
| <b>-B</b> |    |    |    |    | .  | \$ | ,  | #  | »  | <sup>o</sup> | ¿  | l  | ô  | û  | Ô  | Û  |
| <b>-C</b> |    |    |    |    | <  | *  | %  | @  | ö  | æ            | Ð  | —  | ö  | ü  | Ö  | Ü  |
| <b>-D</b> |    |    |    |    | (  | )  | _  | '  | ÿ  | ˆ            | Ÿ  | ¨  | ò  | ù  | Ò  | Ù  |
| <b>-E</b> |    |    |    |    | +  | ;  | >  | =  | þ  | Æ            | Ɔ  | '  | ó  | ú  | Ó  | Ú  |
| <b>-F</b> |    |    |    |    | !  | ^  | ?  | "  | ±  | ∅            | ®  | √  | õ  | ÿ  | Õ  | EO |

Figure 32-21. International: EBCDIC Code Page 500

| ↓  | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0 |    |    |    |    | &  | -  | ∨  | ˘  | °  | a  | ·  | {  | }  | \  |    | 0  |
| -1 |    |    |    |    | ?  | /  | ?  | a  | j  | ~  | Å  | À  | J  | ÷  |    | 1  |
| -2 |    |    |    | ?  | é  | ?  | Ë  | b  | k  | s  | ž  | B  | K  | S  |    | 2  |
| -3 |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | Ť  | C  | L  | T  |    | 3  |
| -4 |    |    |    | ‡  | û  | "  | Ů  | d  | m  | u  | Ž  | D  | M  | U  |    | 4  |
| -5 |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | §  | E  | N  | V  |    | 5  |
| -6 |    |    |    | ǎ  | ?  | Ǟ  | ?  | f  | o  | w  | ž  | F  | O  | W  |    | 6  |
| -7 |    |    |    | č  | ǐ  | č  | ǎ  | g  | p  | x  | ž  | G  | P  | X  |    | 7  |
| -8 |    |    |    | ?  | ǐ  | ?  | ǎ  | h  | q  | y  | Ž  | H  | Q  | Y  |    | 8  |
| -9 |    |    |    | č  | ?  | č  | '  | i  | r  | z  | Ž  | I  | R  | Z  |    | 9  |
| -A |    |    |    | [  | ]  |    | :  | ś  | ł  | Ś  | Ł  | ?  | ě  | ď  | Ď  |    |
| -B |    |    |    | .  | \$ | ,  | #  | ň  | ń  | Ń  | Ń  | ?  | ü  | ?  | Ü  |    |
| -C |    |    |    | <  | *  | %  | @  | đ  | š  | Đ  | Š  | ?  | ?  | ?  | ?  |    |
| -D |    |    |    | (  | )  | _  | '  | ý  | „  | Ý  | „  | ı  | ť  | Ř  | Ť  |    |
| -E |    |    |    | +  | ;  | >  | =  | ř  | „  | Ř  | „  | ?  | ?  | ?  | ?  |    |
| -F |    |    |    | !  | ^  | ?  | "  | ş  | œ  | Ş  | x  | ö  | ě  | Ö  |    |    |

Figure 32-22. Czechoslovakia/Hungary/Poland/Yugoslavia: EBCDIC Code Page 870

| →<br>↓ | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9-           | A- | B- | C- | D- | E- | F- |
|--------|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| -0     |    |    |    |    |    | &  | -  | ?  | ?  | °            | μ  | ?  | þ  | ?  | '  | 0  |
| -1     |    |    |    |    |    | ?  | /  | ?  | a  | j            | ?  | ?  | A  | J  | ÷  | 1  |
| -2     |    |    |    |    | ?  | ?  | ?  | ?  | b  | k            | s  | ¥  | B  | K  | S  | 2  |
| -3     |    |    |    |    | ?  | ?  | ?  | ?  | c  | l            | t  | ·  | C  | L  | T  | 3  |
| -4     |    |    |    |    | ?  | ?  | ?  | ?  | d  | m            | u  | ©  | D  | M  | U  | 4  |
| -5     |    |    |    |    | ?  | ?  | ?  | ?  | e  | n            | v  | §  | E  | N  | V  | 5  |
| -6     |    |    |    |    | ?  | ?  | ?  | ?  | f  | o            | w  | ¶  | F  | O  | W  | 6  |
| -7     |    |    |    |    | ?  | ?  | ?  | ?  | g  | p            | x  | ¼  | G  | P  | X  | 7  |
| -8     |    |    |    |    | ?  | ?  | ?  | ?  | h  | q            | y  | ½  | H  | Q  | Y  | 8  |
| -9     |    |    |    |    | ?  | ?  | ?  | ð  | i  | r            | z  | ¾  | I  | R  | Z  | 9  |
| -A     |    |    |    |    | Ð  | ?  | !  | :  | ?  | <sup>a</sup> | ?  | ⌋  | ?  | 1  | 2  | 3  |
| -B     |    |    |    |    | .  | \$ | ,  | #  | ?  | º            | ?  |    | ?  | ?  | ?  | ?  |
| -C     |    |    |    |    | <  | *  | %  | Ð  | \  | }            | @  | —  | ~  | ?  | ^  | ]  |
| -D     |    |    |    |    | (  | )  | _  | '  | ý  | ˆ            | ÿ  | ¨  | ?  | ?  | ?  | ?  |
| -E     |    |    |    |    | +  | ;  | >  | =  | {  | }            | [  | \  | ?  | ?  | ?  | ?  |
| -F     |    |    |    |    | !  | ?  | ?  | "  | †  | ⊗            | ®  | x  | ?  | ?  | ?  |    |

Figure 32-23. Iceland: EBCDIC Code Page 871

| →<br>↓ | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0     |    |    |    |    |    | &  | -  | ?  | ?  | °  | μ  | ?  | ?  | ğ  | ?  | 0  |
| -1     |    |    |    |    |    | ?  | /  | ?  | a  | j  | ?  | ?  | A  | J  | ÷  | 1  |
| -2     |    |    |    |    | ?  | ?  | ?  | ?  | b  | k  | s  | ¥  | B  | K  | S  | 2  |
| -3     |    |    |    |    | ?  | ?  | ?  | ?  | c  | l  | t  | ·  | C  | L  | T  | 3  |
| -4     |    |    |    |    | ?  | ?  | ?  | ?  | d  | m  | u  | ©  | D  | M  | U  | 4  |
| -5     |    |    |    |    | ?  | ?  | ?  | ?  | e  | n  | v  | §  | E  | N  | V  | 5  |
| -6     |    |    |    |    | ?  | ?  | ?  | ?  | f  | o  | w  | ¶  | F  | O  | W  | 6  |
| -7     |    |    |    |    | ?  | ?  | ?  | ?  | g  | p  | x  | ¼  | G  | P  | X  | 7  |
| -8     |    |    |    |    | {  | ?  | [  | ?  | h  | q  | y  | ½  | H  | Q  | Y  | 8  |
| -9     |    |    |    |    | ?  | ?  | ?  | 1  | i  | r  | z  | ¾  | I  | R  | Z  | 9  |
| -A     |    |    |    |    | ?  | Ğ  | §  | :  | ?  | ª  | ?  | ¬  | ?  | 1  | 2  | 3  |
| -B     |    |    |    |    | .  | ı  | ,  | ?  | ?  | º  | ?  |    | ?  | ?  | ?  | ?  |
| -C     |    |    |    |    | <  | *  | %  | §  | }  | ?  | ]  | —  | ~  | \  | #  | "  |
| -D     |    |    |    |    | (  | )  | _  | '  | `  | ˆ  | \$ | ¨  | ?  | ?  | ?  | ?  |
| -E     |    |    |    |    | +  | ;  | >  | =  | !  | ?  | @  | '  | ?  | ?  | ?  | ?  |
| -F     |    |    |    |    | !  | ^  | ?  | ?  | ‡  | ⊗  | ®  | x  | ?  | ?  | ?  |    |

Figure 32-24. Turkey: EBCDIC Code Page 1026

---

## Appendix A. Data Types

The following data types are used in Presentation Manager. They are listed in alphabetic order.

---

### ACCEL

Accelerator structure.

#### Syntax

```
typedef struct _ACCEL {
    USHORT      fs;
    USHORT      key;
    USHORT      cmd;
} ACCEL;

typedef ACCEL *PACCEL;
```

#### Fields

**fs** (USHORT)

Options.

**key** (USHORT)

Key.

**cmd** (USHORT)

Command code.

The value to be placed in the *uscmd* parameter of a WM\_HELP, a WM\_COMMAND, or a WM\_SYSCOMMAND.

---

### ACCELTABLE

Accelerator-table structure.

#### Syntax

```
typedef struct _ACCELTABLE {
    USHORT      cAccel;
    USHORT      codepage;
    ACCEL       accel[1];
} ACCELTABLE;

typedef ACCELTABLE *PACCELTABLE;
```



## Fields

### cAccel (USHORT)

Number of accelerator entries.

### codepage (USHORT)

Code page for accelerator entries.

### aaccel[1] (ACCEL)

Accelerator entries.

The default accelerator table has the following 16 entries:

| Options            |            | Key        | Command        |
|--------------------|------------|------------|----------------|
| HELP               | VIRTUALKEY | VK_F1      | 0              |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F4      | SC_CLOSE       |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_ENTER   | SC_RESTORE     |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_NEWLINE | SC_RESTORE     |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F5      | SC_RESTORE     |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F6      | SC_NEXTFRAME   |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F7      | SC_MOVE        |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F8      | SC_SIZE        |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F9      | SC_MINIMIZE    |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_F10     | SC_MAXIMIZE    |
| SYSCOMMAND         | VIRTUALKEY | VK_F10     | SC_APPMENU     |
| SYSCOMMAND LONEKEY | VIRTUALKEY | VK_ALT     | SC_APPMENU     |
| SYSCOMMAND LONEKEY | VIRTUALKEY | VK_ALTGRAF | SC_APPMENU     |
| SYSCOMMAND ALT     | VIRTUALKEY | VK_SPACE   | SC_SYSTEMU     |
| SYSCOMMAND SHIFT   | VIRTUALKEY | VK_ESC     | SC_SYSTEMU     |
| SYSCOMMAND CONTROL | VIRTUALKEY | VK_ESC     | SC_TASKMANAGER |

---

## APIRET

Unsigned integer in the range 0 through 4 294 967 295.

### Syntax

```
typedef unsigned long APIRET;
```

---

## APSZ

An array of pointers to NULL-terminated strings.

### Syntax

```
typedef PSZ APSZ[1];
```

---

## ARCPARAMS

Arc-parameters structure.

### Syntax

```
typedef struct _ARCPARAMS {  
    LONG    IP;  
    LONG    IQ;  
    LONG    IR;  
    LONG    IS;  
} ARCPARAMS;  
  
typedef ARCPARAMS *PARCPARAMS;
```

### Fields

#### IP (LONG)

P coefficient.

#### IQ (LONG)

Q coefficient.

#### IR (LONG)

R coefficient.

#### IS (LONG)

S coefficient.

---

## AREABUNDLE

Area-attributes bundle structure.

### Syntax

```
typedef struct _AREABUNDLE {  
    LONG    lColor;  
    LONG    lBackColor;  
    USHORT  usMixMode;  
    USHORT  usBackMixMode;  
    USHORT  usSet;  
    USHORT  usSymbol;  
    POINTL  ptlRefPoint;  
} AREABUNDLE;  
  
typedef AREABUNDLE *PAREABUNDLE;
```

## Fields

### **IColor** (LONG)

Area foreground color.

### **IBackColor** (LONG)

Area background color.

### **usMixMode** (USHORT)

Area foreground-mix mode.

### **usBackColor** (USHORT)

Area background-mix mode.

### **usSet** (USHORT)

Pattern set.

### **usSymbol** (USHORT)

Pattern symbol.

### **ptlRefPoint** (POINTL)

Pattern reference point.

---

## ATOM

Atom identity.

### Syntax

```
typedef ULONG ATOM;
```

---

## BITMAPARRAYFILEHEADER

Bit-map array file header structure.

### Syntax

```
typedef struct _BITMAPARRAYFILEHEADER {  
    USHORT          usType;  
    ULONG           cbSize;  
    ULONG           offNext;  
    USHORT          cxDisplay;  
    USHORT          cyDisplay;  
    BITMAPFILEHEADER bfh;  
} BITMAPARRAYFILEHEADER;  
  
typedef BITMAPARRAYFILEHEADER *PBITMAPARRAYFILEHEADER;
```

## Fields

### **usType** (USHORT)

Type of structure.

Possible values are shown in the following list:

**BFT\_BITMAPARRAY** (0x4142 - 'BA' for BITMAPARRAYFILEHEADER or BITMAPARRAYFILEHEADER2)

### **cbSize** (ULONG)

Size of the BITMAPARRAYFILEHEADER structure in bytes.

### **offNext** (ULONG)

Offset of the next BITMAPARRAYFILEHEADER structure from the start of the file.

### **cxDisplay** (USHORT)

Device width, in pels.

### **cyDisplay** (USHORT)

Device height, in pels.

### **bfh** (BITMAPFILEHEADER)

Bit-map file header structure.

---

## BITMAPARRAYFILEHEADER2

Bit-map array file header structure.

## Syntax

```
typedef struct _BITMAPARRAYFILEHEADER2 {
    USHORT          usType;
    ULONG           cbSize;
    ULONG           offNext;
    USHORT          cxDisplay;
    USHORT          cyDisplay;
    BITMAPFILEHEADER bfh2;
} BITMAPARRAYFILEHEADER2;

typedef BITMAPARRAYFILEHEADER2 *PBITMAPARRAYFILEHEADER2;
```

## Fields

### **usType** (USHORT)

Type of structure.

Possible values are shown in the following list:

**BFT\_BITMAPARRAY** (0x4142. = 'BA' for BITMAPARRAYFILEHEADER or BITMAPARRAYFILEHEADER2)

**cbSize** (ULONG)

Size of the BITMAPARRAYFILEHEADER2 structure in bytes.

**offNext** (ULONG)

Offset of the next BITMAPARRAYFILEHEADER2 structure from the start of the file.

**cxDisplay** (USHORT)

Device width, in pels.

**cyDisplay** (USHORT)

Device height, in pels.

**bmf2** (BITMAPFILEHEADER2)

Bit-map file header structure.

**BITMAPFILEHEADER**

Bit-map file header structure.

**Syntax**

```
typedef struct _BITMAPFILEHEADER {
    USHORT          usType;
    ULONG           cbSize;
    SHORT           xHotspot;
    SHORT           yHotspot;
    USHORT          offBits;
    BITMAPINFOHEADER bmp;
} BITMAPFILEHEADER;

typedef BITMAPFILEHEADER *PBITMAPFILEHEADER;
```

**Fields****usType** (USHORT)

Type of resource the file contains.

Possible values are shown in the following list:

|                  |                                    |
|------------------|------------------------------------|
| BFT_BITMAP       | (0x4D42 - 'BM' for bitmaps)        |
| BFT_ICON         | (0x4349 - 'IC' for icons)          |
| BFT_POINTER      | (0x4540 - 'PT' for pointers)       |
| BFT_COLORICON    | (0x4943 - 'CI' for color icons)    |
| BFT_COLORPOINTER | (0x5043 - 'CP' for color pointers) |

**cbSize** (ULONG)

Size of the BITMAPFILEHEADER structure in bytes.

**xHotspot (SHORT)**

Width of hotspot for icons and pointers.

This field is ignored for bit maps.

**yHotspot (SHORT)**

Height of hotspot for icons and pointers.

This field is ignored for bit maps.

**offBits (USHORT)**

Offset in bytes.

Offset in bytes to beginning of bit-map pel data in the file, from the start of the definition.

**bmp (BITMAPINFOHEADER)**

Bit-map information header structure.

**BITMAPFILEHEADER2**

Bit-map file header structure.

**Syntax**

```
typedef struct _BITMAPFILEHEADER2 {
    USHORT          usType;
    ULONG           cbSize;
    SHORT           xHotspot;
    SHORT           yHotspot;
    USHORT          offBits;
    BITMAPINFOHEADER2 bmp;
} BITMAPFILEHEADER2;

typedef BITMAPFILEHEADER2 *PBITMAPFILEHEADER2;
```

**Fields****usType (USHORT)**

Type of resource the file contains.

Possible values are shown in the following list:

|                  |                                    |
|------------------|------------------------------------|
| BFT_BMAP         | (0x4D42 - 'BM' for bitmaps)        |
| BFT_ICON         | (0x4349 - 'IC' for icons)          |
| BFT_POINTER      | (0x4540 - 'PT' for pointers)       |
| BFT_COLORICON    | (0x4943 - 'CI' for color icons)    |
| BFT_COLORPOINTER | (0x5043 - 'CP' for color pointers) |

**cbSize (ULONG)**

Size of the BITMAPFILEHEADER2 structure in bytes.

**xHotspot (SHORT)**

Width of hotspot for icons and pointers.

This field is ignored for bit maps.

**yHotspot (SHORT)**

Height of hotspot for icons and pointers.

This field is ignored for bit maps.

**offBits (USHORT)**

Offset in bytes.

Offset in bytes to beginning of bit-map pel data in the file, from the start of the definition.

**bmp2 (BITMAPINFOHEADER2)**

Bit-map information header structure.

---

**BITMAPINFO**

Bit-map information structure.

Each bit plane logically contains ( $cx * cy * cBitCount$ ) bits, although the actual length can be greater because of padding.

See also "BITMAPINFO2" on page A-9, which is preferred.

**Syntax**

```
typedef struct _BITMAPINFO {
    ULONG      cbFix;
    USHORT     cx;
    USHORT     cy;
    USHORT     cPlanes;
    USHORT     cBitCount;
    RGB        argbColor[1];
} BITMAPINFO;

typedef BITMAPINFO *PBITMAPINFO;
```

**Fields****cbFix (ULONG)**

Length of fixed portion of structure.

This length can be determined using `sizeof(BITMAPINFOHEADER)`.

**cx (USHORT)**

Bit-map width in pels.

**cy** (USHORT)

Bit-map height in pels.

**cPlanes** (USHORT)

Number of bit planes.

**cBitCount** (USHORT)

Number of bits per pel within a plane.

**argbColor[1]** (RGB)

Array of RGB values.

This is a packed array of 24-bit RGB values. If there are N bits per pel ( $N = cPlanes * cBitCount$ ), the array contains  $2^{**}N$  RGB values. However, if  $N = 24$ , the bit map does not need the *color color* array because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values.

---

## BITMAPINFO2

Bit-map information structure.

Each bit plane logically contains ( $cx * cy * cBitCount$ ) bits, although the actual length can be greater because of padding.

**Note:** Many functions can accept either this structure or the BITMAPINFO structure. Where possible, BITMAPINFO2 should be used.

The *cbFix* field is used to find the color table, if any, that goes with the information in this structure. A color table is an array of color (RGB2) values. If there are N bits per pel ( $N = cPlanes * cBitCount$ ), the array contains  $2^{**}N$  color values. However, if  $N = 24$ , the color table is not required (because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values), unless either *cclrUsed* or *cclrImportant* is non-zero.



## Syntax

```
typedef struct _BITMAPINFO2 {
    ULONG        cbFix;
    ULONG        cx;
    ULONG        cy;
    USHORT       cPlanes;
    USHORT       cBitCount;
    ULONG        ulCompression;
    ULONG        cbImage;
    ULONG        cxResolution;
    ULONG        cyResolution;
    ULONG        cClrUsed;
    ULONG        cClrImportant;
    USHORT       usUnits;
    USHORT       usReserved;
    USHORT       usRecording;
    USHORT       usRendering;
    ULONG        cSize1;
    ULONG        cSize2;
    ULONG        ulColorEncoding;
    ULONG        ulIdentifier;
    RGB2         argbColor[1];
} BITMAPINFO2;

typedef BITMAPINFO2 *PBITMAPINFO2;
```

## Fields

### **cbFix** (ULONG)

Length of fixed portion of structure.

The structure can be truncated after *cBitCount* or any subsequent field.

The length does not include the length of the color table. Where the color table is present, it is at an offset of *cbFix* from the start of the BITMAPINFO2 structure.

This length can range from 16 (BITMAPINFOHEADER through field *cBitCount*) up to sizeof (BITMAPINFOHEADER2) bytes.

### **cx** (ULONG)

Bit-map width in pels.

### **cy** (ULONG)

Bit-map height in pels.

### **cPlanes** (USHORT)

Number of bit planes.

### **cBitCount** (USHORT)

Number of bits per pel within a plane.

### **ulCompression** (ULONG)

Compression scheme used to store the bit map.

### **BCA\_UNCOMP**

Bit map is uncompressed.

### **BCA\_HUFFMAN1D**

The bit map is compressed by a modified Huffman encoding. This is valid for a bi-level (one bit per pel) bit map.

### **BCA\_RLE4**

The bit map is a 4-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

### **BCA\_RLE8**

The bit map is an 8-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

### **BCA\_RLE24**

The bit map is a 24-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

## **Format of Compressed Data**

### **Encoding a run length:**

Run-length encoded bit maps are encoded in the buffer in a controlled format. In all cases, if the first byte is non-zero, it is the length of a run of pels of a particular color or, in the case of a BCA\_RLE4 bit map, a run of a length of pels of alternating colors.

|               |   |
|---------------|---|
| 1st-byte      | pel repetition count $\geq 1$   |
| 2nd-4th bytes | (BCA_RLE24 only) RGB value of pel.  |
| 2nd-byte      | (BCA_RLE8) color index of pel to be repeated<br>(BCA_RLE4) the second byte contains 2 4-bit color indexes. The repetition count is completed by alternately choosing the high-order nibble followed by the low-order nibble for the succeeding pels until the count is exhausted. |

### **Unencoded run:**

An unencoded run is a string of pels to be placed in consecutive positions in the destination bit map.

|          |  |
|----------|--|
| 1st-byte | 0  |
| 2nd-byte | COUNT = a multiple of 3 for BCA_RLE24 bit maps, or<br>COUNT $\geq 3$ (for BCA_RLE4 and BCA_RLE8 bit maps). |

followed by the bytes as follows:

**BCA\_RLE24**

A string of bytes specifying the RGB color values of succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

**BCA\_RLE8**

A string of bytes specifying color indexes for succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

**BCA\_RLE4**

A string of bytes, each byte providing two color indexes, with the high-order nibble specifying the index of the pel preceding the low-order nibble. The COUNT specifies the number of indexes. The overall length of the string must be an even number of bytes, and thus may be padded with a zero byte, and the low order nibble of the last significant byte may also be zero and not used.

**Delta record:**

A delta record indicates a shift in position in the destination bit map before the next record is decoded.

|          |                    |
|----------|--------------------|
| 1st-byte | 0                  |
| 2nd-byte | 2                  |
| 3rd-byte | Delta-x (unsigned) |
| 4th-byte | Delta-y (unsigned) |

This is a relative jump record. It implies that the next record is to be decoded into a position in the destination bit map at an offset from the current position, determined by changing the horizontal and vertical positions by Delta-x and Delta-y, respectively.

**End-of-line record:** The end-of-line record signifies that the data for the current scan line is complete and that decoding of the next record should begin at the start of the next scan line.

|          |   |
|----------|---|
| 1st-byte | 0 |
| 2nd-byte | 0 |

**End-of-RLE record:**

The end-of-RLE record signifies the end of the data in the RLE compressed bit map.

|          |   |
|----------|---|
| 1st-byte | 0 |
| 2nd-byte | 1 |

**cblmage (ULONG)**

Length of bit-map storage data, in bytes.

If the bit map is uncompressed, zero (default) can be specified for this.

**cxResolution (ULONG)**

Horizontal component of the resolution of target device.

The resolution of the device the bit map is intended for, in the units specified by *usUnits*. This information enables an application to select from a resource group the bit map that best matches the characteristics of the current output device.

**cyResolution** (ULONG)

Vertical component of the resolution of the target device.

See the description of *cxResolution*.

**cclrUsed** (ULONG)

Number of color indexes used.

The number of color indexes from the color table that are used by the bit map. If it is zero (the default), all the indexes are used. If it is non-zero, only the first *cclrUsed* entries in the table are accessed by the system, and further entries can be omitted.

For the standard formats with a *cBitCount* of 1, 4, or 8 (and *cPlanes* equal to 1), any indexes beyond *cclrUsed* are not valid. For example, a bit map with 64 colors can use the 8-bitcount format without having to supply the other 192 entries in the color table. For the 24-bitcount standard format, *cclrUsed* is the number of colors used by the bit map.

**cclrImportant** (ULONG)

Minimum number of color indexes for satisfactory appearance of the bit map.

More colors may be used in the bit map, but it is not necessary to assign them to the device palette. These additional colors may be mapped to the nearest colors available.

Zero (the default) means that all entries are important.

For a 24-bitcount standard format bit map, the *cclrImportant* colors are also listed in the color table following the BITMAPINFO2 structure.

**usUnits** (USHORT)

Units of measure.

Units of measure of the horizontal and vertical components of resolution, *cxResolution* and *cyResolution*.

BRU\_METRIC Pels per meter. This is the default value.

**usReserved** (USHORT)

Reserved.

This is a reserved field.

**usRecording** (USHORT)

Recording algorithm.

The format in which the bit map data is recorded.

BRA\_BOTTOMUP Scan lines are recorded bottom to top. This is the default value.

### **usRendering** (USHORT)

Halftoning algorithm.

The algorithm used to record bit map data that has been digitally halftoned.

|                    |   |
|--------------------|---|
| BRH_NOTHALFTONED   | Bit-map data is not halftoned. This is the default value. |
| BRH_ERRORDIFFUSION | Error Diffusion or Damped Error Diffusion algorithm.      |
| BRH_PANDA          | Processing Algorithm for Non-coded Document Acquisition.  |
| BRH_SUPERCIRCLE    | Super Circle algorithm.                                   |

### **cSize1** (ULONG)

Size value 1.

If BRH\_ERRORDIFFUSION is specified in *usRendering*, *cSize1* is the error damping as a percentage in the range 0 through 100. A value of 100% indicates no damping, and a value of 0% indicates that any errors are not diffused.

If BRH\_PANDA or BRH\_SUPERCIRCLE is specified, *cSize1* is the x dimension of the pattern used, in pels.

### **cSize2** (ULONG)

Size value 2.

If BRH\_ERRORDIFFUSION is specified in *usRendering*, this parameter is ignored.

If BRH\_PANDA or BRH\_SUPERCIRCLE is specified, *cSize2* is the y dimension of the pattern used, in pels.

### **ulColorEncoding** (ULONG)

Color encoding.

BCE\_RGB Each element in the color array is an RGB2 datatype. This is the default value.

### **ulIdentifier** (ULONG)

Reserved for application use.

### **argbColor[1]** (RGB2)

Array of RGB values.

This is a packed array of 24-bit RGB values. If there are N bits per pel ( $N = cPlanes * cBitCount$ ), the array contains  $2^{**}N$  RGB values. However, if  $N = 24$ , the bit map does not need the *color* array because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values.

---

## **BITMAPINFOHEADER**

Bit-map information header structure.

Each bit plane logically contains ( $cx * cy * cBitCount$ ) bits, although the actual length can be greater because of padding.

See also BITMAPINFOHEADER2, which is preferred.

## Syntax

```
typedef struct _BITMAPINFOHEADER {
    ULONG      cbFix;
    USHORT     cx;
    USHORT     cy;
    USHORT     cPlanes;
    USHORT     cBitCount;
} BITMAPINFOHEADER;

typedef BITMAPINFOHEADER *PBITMAPINFOHEADER;
```

### Fields

**cbFix** (ULONG)

Length of structure.

**cx** (USHORT)

Bit-map width in pels.

**cy** (USHORT)

Bit-map height in pels.

**cPlanes** (USHORT)

Number of bit planes.

**cBitCount** (USHORT)

Number of bits per pel within a plane.

---

## BITMAPINFOHEADER2

Bit-map information header structure.

Each bit plane logically contains ( $cx * cy * cBitCount$ ) bits, although the actual length can be greater because of padding.

**Note:** Many functions can accept either this structure or the BITMAPINFOHEADER structure. Where possible, use BITMAPINFOHEADER2.

## Syntax

```
typedef struct BITMAPINFOHEADER2 {
    ULONG      cbFix;
    ULONG      cx;
    ULONG      cy;
    USHORT     cPlanes;
    USHORT     cBitCount;
    ULONG      ulCompression;
    ULONG      cbImage;
    ULONG      cxResolution;
    ULONG      cyResolution;
    ULONG      cClrUsed;
    ULONG      cClrImportant;
    USHORT     usUnits;
    USHORT     usReserved;
    USHORT     usRecording;
    USHORT     usRendering;
    ULONG      cSize1;
    ULONG      cSize2;
    ULONG      ulColorEncoding;
    ULONG      ulIdentifier;
} BITMAPINFOHEADER2;

typedef BITMAPINFOHEADER2 *PBITMAPINFOHEADER2;
```

### Fields

#### **cbFix** (ULONG)

Length of structure.

The structure can be truncated after *cBitCount* or any subsequent field.

#### **cx** (ULONG)

Bit-map width in pels.

#### **cy** (ULONG)

Bit-map height in pels.

#### **cPlanes** (USHORT)

Number of bit planes.

#### **cBitCount** (USHORT)

Number of bits per pel within a plane.

#### **ulCompression** (ULONG)

Compression scheme used to store the bit map.

#### **BCA\_UNCOMP**

Bit map is uncompressed.

#### **BCA\_HUFFMAN1D**

The bit map is compressed by a modified Huffman encoding. This is valid for a bi-level (one bit per pel) bit map.

### **BCA\_RLE4**

The bit map is a 4-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

### **BCA\_RLE8**

The bit map is an 8-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

### **BCA\_RLE24**

The bit map is a 24-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

## **Format of Compressed Data**

### **Encoding a run length:**

Run length encoded bit maps are encoded in the buffer in a controlled format. In all cases, if the first byte is non-zero, it is the length of a run of pels of a particular color or, in the case of a BCA\_RLE4 bit map, a run of a length of pels of alternating colors.

|               |   |
|---------------|---|
| 1st-byte      | pel repetition count $\geq 1$   |
| 2nd-4th bytes | (BCA_RLE24 only) RGB value of pel.  |
| 2nd-byte      | (BCA_RLE8) color index of pel to be repeated  |
|               | (BCA_RLE4) the second byte contains 2 4-bit color indexes. The repetition count is completed by alternately choosing the high-order nibble followed by the low-order nibble for the succeeding pels until the count is exhausted. |

### **Unencoded run:**

An unencoded run is a string of pels to be placed in consecutive positions in the destination bit map.

|          |  |
|----------|--|
| 1st-byte | 0  |
| 2nd-byte | COUNT = a multiple of 3 for BCA_RLE24 bit maps, or<br>COUNT $\geq 3$ (for BCA_RLE4 and BCA_RLE8 bit maps). |

followed by the bytes as follows:

### **BCA\_RLE24**

A string of bytes specifying the RGB color values of succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

### **BCA\_RLE8**

A string of bytes specifying color indexes for succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

### **BCA\_RLE4**

A string of bytes, each byte providing two color indexes, with the high-order nibble specifying the index of the pel preceding the low-order nibble. The COUNT specifies the number of indexes. The overall length of the string must be an even number of



bytes, and thus may be padded with a zero byte, and the low order nibble of the last significant byte may also be zero and not used.

**Delta record:**

A delta record indicates a shift in position in the destination bit map before the next record is decoded.

|          |                    |
|----------|--------------------|
| 1st-byte | 0                  |
| 2nd-byte | 2                  |
| 3rd-byte | Delta-x (unsigned) |
| 4th-byte | Delta-y (unsigned) |

This is a relative jump record. It implies that the next record is to be decoded into a position in the destination bit map at an offset from the current position, determined by changing the horizontal and vertical positions by Delta-x and Delta-y, respectively.

**End-of-line record:**

The end-of-line record signifies that the data for the current scan line is complete and that decoding of the next record should begin at the start of the next scan line.

|          |   |
|----------|---|
| 1st-byte | 0 |
| 2nd-byte | 0 |

**End-of-RLE record:**

The end-of-RLE record signifies the end of the data in the RLE compressed bit map.

|          |   |
|----------|---|
| 1st-byte | 0 |
| 2nd-byte | 1 |

**cbImage (ULONG)**

Length of bit-map storage data, in bytes.

If the bit map is uncompressed, zero (the default) can be specified for this.

**cxResolution (ULONG)**

Horizontal component of the resolution of target device.

The resolution of the device the bit map is intended for, in the units specified by *usUnits*. This information enables applications to select from a resource group the bit map that best matches the characteristics of the current output device.

**cyResolution (ULONG)**

Vertical component of the resolution of target device.

See the description of *cxResolution*.

**cclrUsed (ULONG)**

Number of color indexes used.

The number of color indexes from the color table that are used by the bit map. If this is zero (the default), all the indexes are used. If it is non-zero, only the first *cclrUsed* entries in the table are accessed by the system, and further entries can be omitted.

For the standard formats with a *cBitCount* of 1, 4, or 8 (and *cPlanes* equal to 1), any indexes beyond *cclrUsed* are invalid. For example, a bit map with 64 colors can use the

8-bitcount format without having to supply the other 192 entries in the color table. For the 24-bitcount standard format, *cclrUsed* is the number of colors used by the bit map.

**cclrImportant** (ULONG)

Minimum number of color indexes for satisfactory appearance of the bit map.

More colors may be used in the bit map, but it is not necessary to assign them to the device palette. These additional colors may be mapped to the nearest colors available.

Zero (the default) means that all entries are important.

For a 24-bitcount standard format bit map, the *cclrImportant* colors are also listed in the color table relating to this bit map.

**usUnits** (USHORT)

Units of measure.

Units of measure of the horizontal and vertical resolution, *cxResolution* and *cyResolution*.

**BRU\_METRIC** Pels per meter. This is the default value.

**usReserved** (USHORT)

Reserved.

This is a reserved field. If present, it must be zero.

**usRecording** (USHORT)

Recording algorithm.

The format in which the bit-map data is recorded.

**BRA\_BOTTOMUP** Scan lines are recorded bottom to top. This is the default value.

**usRendering** (USHORT)

Halftoning algorithm.

The algorithm used to record bit-map data that has been digitally halftoned.

**BRH\_NOTHALFTONED** Bit-map data is not halftoned. This is the default value.

**BRH\_ERRORDIFFUSION** Error Diffusion or Damped Error Diffusion algorithm.

**BRH\_PANDA** Processing Algorithm for Non-coded Document Acquisition.

**BRH\_SUPERCIRCLE** Super Circle algorithm.

**cSize1** (ULONG)

Size value 1.

If **BRH\_ERRORDIFFUSION** is specified in *usRendering*, *cSize1* is the error damping as a percentage in the range 0 through 100. A value of 100% indicates no damping, and a value of 0% indicates that any errors are not diffused.

If **BRH\_PANDA** or **BRH\_SUPERCIRCLE** is specified, *cSize1* is the x dimension of the pattern used, in pels.

**cSize2** (ULONG)

Size value 2.

If **BRH\_ERRORDIFFUSION** is specified in *usRendering*, this parameter is ignored.

If BRH\_PANDA or BRH\_SUPERCIRCLE is specified, *cSize2* is the y dimension of the pattern used, in pels.

**ulColorEncoding (ULONG)**

Color encoding.

BCE\_RGB Each element in the color array is an RGB2 datatype. This is the default value.

**ullIdentifier (ULONG)**

Reserved for application use.

---

**BIT16**

Defines 16 independent BOOL values.

**Syntax**

```
typedef use BIT16;
```

---

**BIT32**

Defines 32 independent BOOL values.

**Syntax**

```
typedef use BIT32;
```

---

**BIT8**

Defines eight independent BOOL values.

**Syntax**

```
typedef use BIT8;
```

---

## BOOL

Boolean.

Valid values are FALSE, which is 0, and TRUE, which is 1.

### Syntax

```
typedef unsigned long BOOL;
```

---

## BOOKPAGEINFO

Notebook page information structure.

### Syntax

```
typedef struct _BOOKPAGEINFO {
    ULONG          cb;
    ULONG          fl;
    BOOL           bLoadDlg;
    ULONG          ulPageData;
    HWND          hwndPage;
    PFN           pfnPageDlgProc;
    ULONG          idPageDlg;
    HMODULE        hmodPageDlg;
    PVOID          pPageDlgCreateParam;
    PDLGTEMPLATE   pdlgtPage;
    ULONG          cbStatusLine;
    PSZ            pszStatusLine;
    HBITMAP        hbmMajorTab;
    HBITMAP        hbmMinorTab;
    ULONG          cbMajorTab;
    PSZ            pszMajorTab;
    ULONG          cbMinorTab;
    PSZ            pszMinorTab;
    PVOID          pBidiInfo;
} BOOKPAGEINFO;

typedef BOOKPAGEINFO *PBOOKPAGEINFO;
```

### Fields

**cb** (ULONG)

Size of the page information structure.

**fl** (ULONG)

Flag indicating which page attributes are to be set.

BFA\_BIDIINFO

Reserved for bi-directional support.

|                         |  |
|-------------------------|--|
| BFA_MAJORTABBITMAP      | Set/query major tab bit map.                         |
| BFA_MAJORTABTEXT        | Set/query major tab text.                            |
| BFA_MINORTABBITMAP      | Set/query minor tab bit map.                         |
| BFA_MINORTABTEXT        | Set/query minor tab text.                            |
| BFA_PAGEDATA            | Set/query page data.                                 |
| BFA_PAGEFROMDLGRES      | Set/query page window handle from a dialog resource. |
| BFA_PAGEFROMDLGTEMPLATE | Set/query page window handle from a dialog template. |
| BFA_PAGEFROMHWND        | Set/query page window handle.                        |
| BFA_STATUSLINE          | Set/query status text.                               |

**bLoadDlg** (BOOL)

Load dialog flag.

TRUE     Load dialog immediately.  
FALSE     Load dialog on page turn.

**ulPageData** (ULONG)

Data to associate with the notebook page.

**hwndPage** (HWND)

Handle to associate with the notebook page.

**pfnPageDlgProc** (PFN)

Dialog procedure.

**idPageDlg** (ULONG)

Dialog id.

**hmodPageDlg** (HMODULE)

Resource handle.

**pPageDlgCreateParam** (PVOID)

Dialog create parameters.

**pdlgPage** (PDLGTEMPLATE)

Dialog template.

**cbStatusLine** (ULONG)

Length of status line text.

**pszStatusLine** (PSZ)

Status line text string.

**hbmMajorTab** (HBITMAP)

Major tab bit map handle.

**hbmMinorTab** (HBITMAP)

Minor tab bit map handle.

**cbMajorTab** (ULONG)

Length of major tab text.

**pszMajorTab** (PSZ)

Major tab text string.

**cbMinorTab** (ULONG)

Length of minor tab text.

**pszMinorTab** (PSZ)

Minor tab text string.

**pBidilInfo** (PVOID)

Reserved for bi-directional support.

---

## BOOKTEXT

Notebook data structure that contains text strings for notebook status lines and tabs. This data structure is used with the BKM\_QUERYSTATUSLINETEXT and the BKM\_QUERYTABTEXT messages only. See "BKM\_QUERYSTATUSLINETEXT" on page 23-18 and "BKM\_QUERYTABTEXT" on page 23-20 for information about those messages.

### Syntax

```
typedef struct _BOOKTEXT {
    PSZ      pString;
    ULONG    textLen;
} BOOKTEXT;

typedef BOOKTEXT *PBOOKTEXT;
```

### Fields

**pString** (PSZ)

Pointer to a string buffer.

Buffer in which the text string is to be placed. For the BKM\_QUERYSTATUSLINETEXT message, this is the buffer in which the status line text is placed.

For the BKM\_QUERYTABTEXT message, this is the buffer in which the tab text is placed.

**textLen** (ULONG)

String length.

Length of the text string. For the BKM\_QUERYSTATUSLINETEXT message, this is the length of the status line text string.

For the BKM\_QUERYTABTEXT message, this is the length of the tab text string.

---

## BTNCDATA

Button-control-data structure.

### Syntax

```
typedef struct _BTNCDATA {
    USHORT      cb;
    USHORT      fsCheckState;
    USHORT      fsHiliteState;
    LHANDLE     hImage;
} BTNCDATA;

typedef BTNCDATA *PBTNCDATA;
```

### Fields

#### cb (USHORT)

Length of the control data in bytes.

This is the length of the control data for a button control.

#### fsCheckState (USHORT)

Check state of button.

This is the same value as returned by the BM\_QUERYCHECK message and passed to the BM\_SETCHECK message.

#### fsHiliteState (USHORT)

Highlighting state of button.

This is the same value as returned by the BM\_QUERYHILITE message and passed to the BM\_SETHILITE message.

#### hImage (LHANDLE)

Resource handle for icon or bit map.

---

## BYTE

A byte.

### Syntax

```
typedef unsigned char BYTE;
```

---

## CATCHBUF

Saved execution environment buffer.

### Syntax

```
typedef struct _CATCHBUF {
    ULONG     reserved[4];
} CATCHBUF;

typedef CATCHBUF *PCATCHBUF;
```

### Fields

**reserved[4]** (ULONG)

Save area.

---

## CDATE

Structure that contains date information for a data element in the details view of a container control.

### Syntax

```
typedef struct _CDATE {
    UCHAR     day;
    UCHAR     month;
    USHORT    year;
} CDATE;

typedef CDATE *PCDATE;
```

### Fields

**day** (UCHAR)

Current day.

**month** (UCHAR)

Current month.

**year** (USHORT)

Current year.



---

## CHAR

Single-byte character.

### Syntax

```
#define CHAR char
```

---

## CHARBUNDLE

Character-attributes bundle structure.

### Syntax

```
typedef struct _CHARBUNDLE {  
    LONG        lColor;  
    LONG        lBackColor;  
    USHORT     usMixMode;  
    USHORT     usBackMixMode;  
    USHORT     usSet;  
    USHORT     usPrecision;  
    SIZEF      sizfxCell;  
    POINTL     ptlAngle;  
    POINTL     ptlShear;  
    USHORT     usDirection;  
    USHORT     usTextAlign;  
    FIXED      fxExtra;  
    FIXED      fxBreakExtra;  
} CHARBUNDLE;  
  
typedef CHARBUNDLE *PCHARBUNDLE;
```

### Fields

#### **IColor** (LONG)

Character foreground color.

#### **IBackColor** (LONG)

Character background color.

#### **usMixMode** (USHORT)

Character foreground-mix mode.

#### **usBackMixMode** (USHORT)

Character background-mix mode.

#### **usSet** (USHORT)

Character set.

**usPrecision** (USHORT)

Character precision.

**sizfCell** (SIZEF)

Character cell size.

**ptlAngle** (POINTL)

Character angle.

**ptlShear** (POINTL)

Character shear.

**usDirection** (USHORT)

Character direction.

**usTextAlign** (USHORT)

Text alignment.

**fxExtra** (FIXED)

Character extra.

**fxBreakExtra** (FIXED)

Character break extra.

---

## CLASSINFO

Class-information structure.

### Syntax

```
typedef struct _CLASSINFO {
    ULONG      fClassStyle;
    PFNWP     pfnWindowProc;
    ULONG     cbWindowData;
} CLASSINFO;

typedef CLASSINFO *PCLASSINFO;
```

### Fields

**fClassStyle** (ULONG)

Class-style flags.

**pfnWindowProc** (PFNWP)

Window procedure.

**cbWindowData** (ULONG)

Number of additional window words.

---

## CNRDRAGINFO

Structure that contains information about a direct manipulation event that is occurring over the container. The information specified for this structure depends on the container notification code with which it is used. The differences are specified in the following field descriptions. The applicable notification codes are:

- “CN\_DRAGAFTER” on page 22-12
- “CN\_DRAGLEAVE” on page 22-15
- “CN\_DRAGOVER” on page 22-16
- “CN\_DROP” on page 22-18
- “CN\_DROPHELP” on page 22-19

### Syntax

```
typedef struct _CNRDRAGINFO {
    PDRAGINFO      pDragInfo;
    RECORDCORE     pRecord;
} CNRDRAGINFO;

typedef CNRDRAGINFO *PCNRDRAGINFO;
```

### Fields

#### **pDragInfo** (PDRAGINFO)

Pointer to a DRAGINFO structure.

#### **pRecord** (RECORDCORE)

Pointer to a RECORDCORE structure.

The structure that is pointed to depends on the notification code being used.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages. For the CN\_DRAGAFTER notification code, this field contains a pointer to the RECORDCORE structure after which ordered target emphasis is drawn. If ordered target emphasis is applied above the first record in item order, the CM\_FIRST attribute is returned.

For the CN\_DRAGLEAVE notification code, this field is NULL.

For the CN\_DRAGOVER, CN\_DROP, and CN\_DROPHELP notification codes, this field contains a pointer to a container record over which direct manipulation occurred. This field has a value of NULL if the direct manipulation event occurs over white space.

---

## CNRDRAWITEMINFO

Structure that contains information about the container item being drawn. This structure is used with the WM\_DRAWITEM (in Container Controls) message only. See “WM\_DRAWITEM (in Container Controls)” on page 22-7 for information about that message.

## Syntax

```
typedef struct _CNRDRAWITEMINFO {
    RECORDCORE      pRecord;
    PFIELDINFO      pFieldInfo;
} CNRDRAWITEMINFO;

typedef CNRDRAWITEMINFO *PCNRDRAWITEMINFO;
```

## Fields

### **pRecord** (RECORDCORE)

Pointer to the RECORDCORE structure for the record being drawn.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages.

### **pFieldInfo** (PFIELDINFO)

Pointer to the FIELDINFO structure for the container column being drawn in the details view.

For all other views, this field is NULL.

---

## CNREDITDATA

Structure that contains information about the direct editing of container text. The information specified for this structure depends on the container notification code or message with which it is used. The differences are specified in the following field descriptions. The applicable notification codes and message are:

- "CN\_BEGINEDIT" on page 22-10
- "CN\_ENDEDIT" on page 22-21
- "CN\_REALLOCPSZ" on page 22-28
- "CM\_OPENEDIT" on page 22-54

## Syntax

```
typedef struct _CNREDITDATA {
    ULONG          cb;
    HWND          hwndCnr;
    RECORDCORE    pRecord;
    PFIELDINFO    pFieldInfo;
    PSZ           *ppszText;
    ULONG         cbText;
    ULONG         id;
} CNREDITDATA;

typedef CNREDITDATA *PCNREDITDATA;
```

### Fields

#### **cb** (ULONG)

Structure size.

The size (in bytes) of the CNREDITDATA data structure.

#### **hwndCnr** (HWND)

Container window handle.

#### **pRecord** (RECORDCORE)

Pointer to a RECORDCORE data structure, or NULL.

This field is NULL if container titles are to be edited.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

For the CN\_BEGINEDIT, CN\_ENDEDIT, and CN\_REALLOCPSZ notification codes, this field is a pointer to the edited RECORDCORE data structure.

For the CM\_OPENEDIT message, this field is a pointer to the RECORDCORE data structure to be edited.

#### **pFieldInfo** (PFIELDINFO)

Pointer to a FIELDINFO data structure, or NULL.

Pointer to a FIELDINFO data structure if the current view is the details view and the user is not editing the container title. Otherwise, this field is NULL.

If the current view is the details view:

- For the CN\_BEGINEDIT, CN\_ENDEDIT, and CN\_REALLOCPSZ notification codes, this field contains a pointer to the FIELDINFO structure being edited.
- For the CM\_OPENEDIT message, this field is a pointer to the FIELDINFO data structure to be edited.

**ppszText (PSZ \*)**

Pointer to a PSZ text string.

For the CN\_BEGINEDIT and CN\_REALLOCPSZ notification codes, this field is a pointer to the current PSZ text string.

For the CN\_ENDEDIT notification code, this field is a pointer to the new PSZ text string.

For the CM\_OPENEDIT message, this field is NULL.

**cbText (ULONG)**

Number of bytes in the text string.

For the CN\_BEGINEDIT notification code, this field is 0.

For the CN\_ENDEDIT and CN\_REALLOCPSZ notification codes, this field is the number of bytes in the new text string.

For the CM\_OPENEDIT message, this field is 0.

**id (ULONG)**

ID of the window to be edited.

The ID can be one of the following:

CID\_CNRTITLEWND

Title window.

CID\_LEFTDVWND

Left details view window; default if unsplit window.

CID\_RIGHTDVWND

Right details view window.

CID\_LEFTCOLTITLWIND

Left details view column headings window; default if unsplit window.

CID\_RIGHTCOLTITLWIND

Right details view column headings window.

*An application-defined container-ID*

Container window.

---

## CNRDRAGINIT

Structure that contains information about a direct manipulation event that is initiated in a container. This structure is used with the CN\_INITDRAG notification code only. See "CN\_INITDRAG" on page 22-24 for information about that notification code.

### Syntax

```
typedef struct _CNRDRAGINIT {
    HWND          hwndCnr;
    RECORDCORE    pRecord;
    LONG          x;
    LONG          y;
    LONG          cx;
    LONG          cy;
} CNRDRAGINIT;

typedef CNRDRAGINIT *PCNRDRAGINIT;
```

### Fields

#### hwndCnr (HWND)

Container control handle.

#### pRecord (RECORDCORE)

Pointer to the RECORDCORE where direct manipulation started.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages.

The *pRecord* field can have one of the following values:

NULL    Direct manipulation started over white space.  
Other    Container record over which direct manipulation started.

#### x (LONG)

X-coordinate of the pointer of the pointing device in desktop coordinates.

#### y (LONG)

Y-coordinate of the pointer of the pointing device in desktop coordinates.

#### cx (LONG)

X-offset from the hot spot of the pointer of the pointing device (in pels) to the record origin.

#### cy (LONG)

Y-offset from the hot spot of the pointer of the pointing device (in pels) to the record origin.

---

## CNRINFO

Structure that contains information about the container.

### Syntax

```
typedef struct _CNRINFO {
    ULONG          cb;
    PVOID          pSortRecord;
    PFIELDINFO     pFieldInfoLast;
    PFIELDINFO     pFieldInfoObject;
    PSZ            pszCnrTitle;
    ULONG          flWindowAttr;
    POINTL         ptlOrigin;
    ULONG          cDelta;
    ULONG          cRecords;
    SIZEL          slBitmapOrIcon;
    SIZEL          slTreeBitmapOrIcon;
    HBITMAP        hbmExpanded;
    HBITMAP        hbmCollapsed;
    HPOINTER       hpPtrExpanded;
    HPOINTER       hpPtrCollapsed;
    LONG           cyLineSpacing;
    LONG           cxTreeIndent;
    LONG           cxTreeLine;
    ULONG          cFields;
    LONG           xVertSplitbar;
} CNRINFO;

typedef CNRINFO *PCNRINFO;
```

### Fields

#### **cb** (ULONG)

Structure size.

The size (in bytes) of the CNRINFO data structure.

#### **pSortRecord** (PVOID)

Pointer to the comparison function for sorting container records, or NULL.

If NULL, which is the default condition, no sorting is performed. Sorting only occurs during record insertion and when changing the value of this field. The third parameter of the comparison function, *pStorage*, must be NULL. See "CM\_SORTRECORD" in the *Presentation Manager Programming Reference* for a further description of the comparison function.

#### **pFieldInfoLast** (PFIELDINFO)

Pointer to last column in the left window of the split details view, or NULL.

The default is NULL, causing all columns to be positioned in the left window.



**pFieldInfoObject (PFIELDINFO)**

Pointer to a column that represents an object in the details view.

The data for this FIELDINFO structure must contain icons or bit maps. In-use emphasis is applied to this column of icons or bit maps only. The default is the leftmost column in the unsplit details view, or the leftmost column in the left window of the split details view.

**pszCnrTitle (PSZ)**

Title text, or NULL.

Text for the container title. The default is NULL.

**flWindowAttr (ULONG)**

Window attributes.

Consists of the following container window attributes:

- Specify one of the following container views, which determine the presentation format of items in a container:

**CV\_ICON**

In the icon view, the container items are represented as icon/text or bit-map/text pairs, with text beneath the icons or bit maps. This is the default view. This view can be combined with the CV\_MINI style bit by using an OR operator (|). See CV\_MINI on page A-35 for more information.

**CV\_NAME**

In the name view, the container items are represented as icon/text or bit-map/text pairs, with text to the right of the icons or bit maps. This view can be combined with the CV\_MINI and CV\_FLOW style bits by using OR operators (|). See CV\_MINI on page A-35 and CV\_FLOW on page A-35 for more information.

**CV\_TEXT**

In the text view, the container items are displayed as a list of text strings. This view can be combined with the CV\_FLOW style bit by using an OR operator (|). See CV\_FLOW on page A-35 for more information.

**CV\_TREE**

In the tree view, the container items are represented in a hierarchical manner. The tree view has three forms, which are defined in the following list. If you specify CV\_TREE by itself, the tree icon view is used.

- Tree icon view

The tree icon view is specified by using a logical OR operator to combine the tree view with the icon view (CV\_TREE | CV\_ICON). Container items in this view are represented as icon/text pairs or bit-map/text pairs, with text to the right of the icons or bit maps. Also, a collapsed or expanded icon or bit map is displayed to the left of parent items. If this icon or bit map is a *collapsed* icon or bit map, selecting it will cause the parent item to be expanded so that its child items are displayed below it. If this icon or bit map is an *expanded* icon or bit map, selecting it will cause the parent's child items to be removed from the display. The default collapsed and

expanded bit maps provided by the container use a plus sign (+) and a minus sign (-), respectively, to indicate that items can be added to or subtracted from the display.

- Tree name view

The tree name view is specified by using a logical OR operator to combine the tree view with the name view (CV\_TREE | CV\_NAME). Container items in this view are displayed as either icon/text pairs or bit-map/text pairs, with text to the right of the icons or bit maps. However, the indicator that represents whether an item can be collapsed or expanded, such as a plus or minus sign, is included in the icon or bit map that represents that item, not in a separate icon or bit map as in the tree icon and tree text views. The container control does not provide default collapsed and expanded bit maps for the tree name view.

- Tree text view

The tree text view is specified by using a logical OR operator to combine the tree view with the text view (CV\_TREE | CV\_TEXT). Container items in this view are displayed as a list of text strings. As in the tree icon view, a collapsed or expanded icon or bit map is displayed to the left of parent items.

### CV\_DETAIL

In the details view, the container items are presented in columns. Each column can contain icons or bit maps, text, numbers, dates, or times.

- Specify one or both of the following view styles by using an OR operator (|) to combine them with the specified view. These view styles are optional.

### CV\_MINI

Produces a mini-icon whose size is based on the Presentation Manager (PM) SV\_CYMENU system value to produce a device-dependent mini-icon.

The CV\_MINI view style bit is ignored when:

- The text view (CV\_TEXT), tree view (CV\_TREE), or details view (CV\_DETAIL) are displayed
- The CCS\_MINIRECORDCORE style bit is specified.

If this style bit is not specified and the icon view (CV\_ICON) or name view (CV\_NAME) is used, the default, regular-sized icon is used. The size of regular-sized icons is based on the value in the *sBitmapOrIcon* field of the CNRINFO data structure. If this field is equal to 0, the PM SV\_CXICON and SV\_CYICON system values for width and height, respectively, are used. Icon sizes are consistent with PM-defined icon sizes for all devices.

### CV\_FLOW

Dynamically arranges container items in columns in the name and text views. These are called flowed name and flowed text views. If this style bit is set for the name view (CV\_NAME) or text view (CV\_TEXT), the container items are placed in a single column until the bottom of the client area is reached. The next container item is placed in the adjacent column to the right of the filled

column. This process is repeated until all of the container items are positioned in the container. The width of each column is determined by the longest text string in that column. The size of the window determines the depth of the client area.

If this style bit is not specified, the default condition for the name and text views is to vertically fill the container in a single column without flowing the container items. If this style bit is set for the icon view (CV\_ICON) or details view (CV\_DETAIL), it is ignored.

- Specify either of the following to indicate whether the container will display icons or bit maps:

#### **CA\_DRAWICON**

Icons are used for the icon, name, tree, or details views. This is the default. This container attribute should be used with the *hptrIcon* and *hptrMiniIcon* fields of the RECORDCORE data structure.

#### **CA\_DRAWBITMAP**

Bit maps are used for the icon, name, tree, or details views. This container attribute can be used with the *hbmBitmap* and *hbmMiniBitmap* fields of the RECORDCORE data structure.

#### **Notes:**

1. If both the CA\_DRAWICON and CA\_DRAWBITMAP attributes are specified, the CA\_DRAWICON attribute is used.
  2. If the CCS\_MINIRECORDCORE style bit is specified when a container is created, the *hptrIcon* field of the MINIRECORDCORE data structure is used.
- Specify one of the following attributes to provide target emphasis for the name, text, and details views. If neither ordered nor mixed target emphasis is specified, the emphasis is drawn around the record.

#### **CA\_ORDEREDTARGETEMPH**

Shows where a container record can be dropped during direct manipulation by drawing a line beneath the record. Ordered target emphasis does not apply to the icon and tree views.

#### **CA\_MIXEDTARGETEMPH**

Shows where a container record can be dropped during direct manipulation either by drawing a line between two items or by drawing lines around the container record. Mixed target emphasis does not apply to the icon and tree views.

- Specify the following attribute to draw lines that show the relationship between items in the tree view.

#### **CA\_TREELINE**

Shows the relationship between all items in the tree view.

- Specify the following to draw container records, paint the background of the container, or both:

**CA\_OWNERDRAW**

**Ownerdraw** for the container, which allows the application to draw container records.

**CA\_OWNERPAINTBACKGROUND**

Allows the application to subclass the container and paint the background. If specified, and the container is subclassed, the application receives the CM\_PAINTBACKGROUND message in the subclass procedure. Otherwise, the container paints the background using the color specified by SYSCLR\_WINDOW, which can be changed by using the PP\_BACKGROUND\_COLOR or PP\_BACKGROUND\_COLORINDEX presentation parameter in the WM\_PRESPARAMCHANGED (in Container Controls)

- Specify the following if the container is to have a title:

**CA\_CONTAINERTITLE**

Allows you to include a container title. The default is no container title.

- Specify one or both of the following container title attributes. These are valid only if the CA\_CONTAINERTITLE attribute is specified.

**CA\_TITLEREADONLY**

Prevents the container title from being edited directly. The default is to allow the container title to be edited.

**CA\_TITLESEPARATOR**

Puts a separator line between the container title and the records beneath it. The default is no separator line.

- Specify one of the following to position the container title. These are valid only if the CA\_CONTAINERTITLE attribute is specified.

**CA\_TITLECENTER**

Centers the container title. This is the default.

**CA\_TITLELEFT**

Left-justifies the container title.

**CA\_TITLERIGHT**

Right-justifies the container title.

- Specify the following to display column headings in the details view:

**CA\_DETAILSVIEWTITLES**

Allows you to include column headings in the details view. The default is no column headings.

**ptlOrigin** (POINTL)

Workspace origin.

Lower-left origin of the workspace in virtual coordinates, used in the icon view. The default origin is **(0,0)**.

**cDelta** (ULONG)

Threshold.

An application-defined threshold, or number of records, from either end of the list of available records. Used when a container needs to handle large amounts of data. The default is 0. Refer to the *OS/2 Programming Guide* for more information about specifying deltas.

**cRecords** (ULONG)

Number of records.

The number of records in the container. Initially this field is 0.

**slBitmapOrIcon** (SIZEL)

Icon/bit-map size.

The size (in pels) of icons or bit maps. The default is the system size.

**slTreeBitmapOrIcon** (SIZEL)

Icon/bit-map size.

The size (in pels) of the expanded and collapsed icons or bit maps used in the tree icon and tree text views.

**hbmExpanded** (HBITMAP)

Bit-map handle.

The handle of the bit map to be used to represent an expanded parent item in the tree icon and tree text views. If neither an icon handle (see *hptrExpanded*) nor a bit-map handle is specified, a default bit map with a minus sign (-) is provided.

**hbmCollapsed** (HBITMAP)

Bit-map handle.

The handle of the bit map to be used to represent a collapsed parent item in the tree icon and tree text views. If neither an icon handle (see *hptrCollapsed*) nor a bit-map handle is specified, a default bit map with a plus sign (+) is provided.

**hptrExpanded** (HPOINTER)

Icon handle.

The handle of the icon to be used to represent an expanded parent item in the tree icon and tree text views. If neither an icon handle nor a bit-map handle (see *hbmExpanded*) is specified, a default bit map with a minus sign (-) is provided.

**hptrCollapsed** (HPOINTER)

Icon handle.

The handle of the icon to be used to represent a collapsed parent item in the tree icon and tree text views. If neither an icon handle nor a bit-map handle (see *hbmCollapsed*) is specified, a default bit map with a plus sign (+) is provided.

**cyLineSpacing** (LONG)

Vertical space.

The amount of vertical space (in pels) between the records. If you specify a value that is less than 0, a default value is used.

**cxTreeIndent** (LONG)

Horizontal space.

The amount of horizontal space (in pels) between levels in the tree view. If you specify a value that is less than 0, a default value is used.

**cxTreeLine** (LONG)

Line width.

The width of the lines (in pels) that show the relationship between tree items. If you specify a value that is less than 0, a default value is used. Also, if the CA\_TREELINE container attribute of the *flWindowAttr* field is not specified, these lines are not drawn.

**cFields** (ULONG)

Number of columns.

The number of FIELDINFO structures in the container. Initially this field is 0.

**xVertSplitbar** (LONG)

Split bar position.

The initial position of the split bar relative to the container, used in the details view. If this value is less than 0, the split bar is not used. The default value is negative one (-1).

---

**CNRLAZYDRAGINFO**

Container lazy drag information.

**Syntax**

```
typedef struct _CNRLAZYDRAGINFO {
    PDRAGINFO      pDragInfo;
    PRECORDCORE    pRecord;
    HWND           hwndTarget;
} CNRLAZYDRAGINFO;

typedef CNRLAZYDRAGINFO *PCNRLAZYDRAGINFO;
```

## Fields

### **pDragInfo** (PDRAGINFO)

Pointer to the DRAGINFO structure.

### **pRecord** (RECORDCORE)

Pointer to a container RECORDCORE structure.

A value of NULL indicates that the lazy drag set was dropped over whitespace in the container. Any other value indicates that the lazy drag set was dropped on the record specified by this field.

### **hwndTarget** (HWND)

Handle of the target window that the lazy drag set was dropped on.

---

## COLOR

Color value.

## Syntax

```
typedef LONG COLOR;
```

---

## CONVCONTEXT

Dynamic-data-exchange conversation context structure.

## Syntax

```
typedef struct _CONVCONTEXT {
    ULONG      cb;
    ULONG      fsContext;
    ULONG      idCountry;
    ULONG      usCodepage;
    ULONG      usLangID;
    ULONG      usSubLangID;
} CONVCONTEXT;

typedef CONVCONTEXT *PCONVCONTEXT;
```

## Fields

### **cb** (ULONG)

Length of structure.

This must be set to the length of the CONVCONTEXT structure.

**fsContext** (ULONG)

Options.

DDEXTXT\_CASESENSITIVE All strings in this conversation are case sensitive.

**idCountry** (ULONG)

Country code.

**usCodepage** (ULONG)

Code-page identity.

**usLangID** (ULONG)

Language.

Zero is valid and means no language information.

**usSubLangID** (ULONG)

Sub-language.

Zero is valid and means no sub-language information.

---

## CREATESTRUCT

Create-window data structure.

### Syntax

```
typedef struct _CREATESTRUCT {
    PVOID      pPresParams;
    PVOID      pCtldata;
    ULONG      id;
    HWND      hwndInsertBehind;
    HWND      hwndOwner;
    LONG      cy;
    LONG      cx;
    LONG      y;
    LONG      x;
    ULONG      flStyle;
    PSZ       pszText;
    PSZ       pszClassName;
    HWND      hwndParent;
} CREATESTRUCT;

typedef CREATESTRUCT *PCREATESTRUCT;
```

### Fields

**pPresParams** (PVOID)

Presentation parameters.

**pCtldata** (PVOID)

Control data.



**id** (ULONG)  
Window identifier.

**hwndInsertBehind** (HWND)  
Window behind which the window is to be placed.

**hwndOwner** (HWND)  
Window owner.

**cy** (LONG)  
Window height.

**cx** (LONG)  
Window width.

**y** (LONG)  
Y-coordinate of origin.

**x** (LONG)  
X-coordinate of origin.

**fiStyle** (ULONG)  
Window style.

**pszText** (PSZ)  
Window text.

**pszClassName** (PSZ)  
Registered window class name.

**hwndParent** (HWND)  
Parent window handle.

---

## CSBITMAPDATA

This is the bit-map data structure for the circular slider buttons.

### Syntax

```
typedef struct _CSBITMAPDATA {
    HBITMAP    hbmLeftUp;
    HBITMAP    hbmLeftDown;
    HBITMAP    hbmRightUp;
    HBITMAP    hbmRightDown;
} CSBITMAPDATA;

typedef CSBITMAPDATA *PCSBITMAPDATA;
```

## Fields

### **hbmLeftUp** (HBITMAP)

Handle to the “up” position bit map for the button on the left.

### **hbmLeftDown** (HBITMAP)

Handle to “down” position bit map for the button on the left.

### **hbmRightUp** (HBITMAP)

Handle to the “up” position bit map for the button on the right.

### **hbmRightDown** (HBITMAP)

Handle to the “down” position bit map for the button on the right.

---

## CURSORINFO

Cursor-information structure.

## Syntax

```
typedef struct _CURSORINFO {
    HWND      hwnd;
    LONG      x;
    LONG      y;
    LONG      cx;
    LONG      cy;
    ULONG     fs;
    RECT      rcClip;
} CURSORINFO;

typedef CURSORINFO *PCURSORINFO;
```

## Fields

### **hwnd** (HWND)

Window handle.

### **x** (LONG)

X-coordinate.

### **y** (LONG)

Y-coordinate.

### **cx** (LONG)

Cursor width.

### **cy** (LONG)

Cursor height.

### **fs** (ULONG)

Options.

**rciClip** (RECTL)  
Cursor box.

---

## CTIME

Structure that contains time information for a data element in the details view of a container control.

### Syntax

```
typedef struct _CTIME {  
    UCHAR    hours;  
    UCHAR    minutes;  
    UCHAR    seconds;  
    UCHAR    ucReserved;  
} CTIME;  
  
typedef CTIME *PCTIME;
```

### Fields

**hours** (UCHAR)  
Current hour.

**minutes** (UCHAR)  
Current minute.

**seconds** (UCHAR)  
Current second.

**ucReserved** (UCHAR)  
Reserved.

---

## Control-Data

Pointer to class-specific control data, beginning with a value conforming to a USHORT data type, which specifies the overall length of the data.

There are several different types of control-data structures:

|                   |                                     |
|-------------------|-------------------------------------|
| <b>BTNCDATA</b>   | Button control data                 |
| <b>ENTRYFDATA</b> | Entry field control data            |
| <b>FRAMECDATA</b> | Frame control data                  |
| <b>MLECTLDATA</b> | Multi-line entry field control data |
| <b>SBCDATA</b>    | Scroll bar control data.            |

---

## DDEINIT

Dynamic-data-exchange initiation structure.

### Syntax

```
typedef struct _DDEINIT {
    ULONG      cb;
    PSZ        pszAppName;
    PSZ        pszTopic;
    ULONG      offConvContext;
} DDEINIT;

typedef DDEINIT *PDDEINIT;
```

### Fields

#### **cb** (ULONG)

Length of structure.

This must be set to the length of the DDEINIT structure.

#### **pszAppName** (PSZ)

Application name.

Pointer to name of the server application.

Application names must not contain slashes or backslashes. These characters are reserved for future use in network implementations.

#### **pszTopic** (PSZ)

Topic.

Pointer to name of the topic.

#### **offConvContext** (ULONG)

Conversation context.

Offset to a CONVCONTEXT structure.

---

## DELETENOTIFY

Structure that contains information about the application page that is being deleted from a notebook.

### Syntax

```
typedef struct _DELETENOTIFY {
    HWND      hwndBook;
    HWND      hwndPage;
    ULONG     ulAppPageData;
    HBITMAP   hbmTab;
} DELETENOTIFY;

typedef DELETENOTIFY *PDELETENOTIFY;
```

### Fields

**hwndBook** (HWND)

Notebook window handle.

**hwndPage** (HWND)

Application page window handle.

**ulAppPageData** (ULONG)

Application-specified page data.

**hbmTab** (HBITMAP)

Application-specified tab bit map.

---

## DDESTRUCT

Dynamic-data-exchange control structure.

### Syntax

```
typedef struct _DDESTRUCT {
    ULONG     cbData;
    USHORT    fsStatus;
    USHORT    usFormat;
    USHORT    offszItemName;
    USHORT    offabData;
} DDESTRUCT;

typedef DDESTRUCT *PDDESTRUCT;
```

## Fields

### **cbData** (ULONG)

Length of the data.

This is the length of data that occurs after the *offabData* parameter. If no data exists, this field should contain a zero (0).

### **fsStatus** (USHORT)

Status of the data exchange.

|                  |  |
|------------------|--|
| DDE_FACK         | Positive acknowledgement   |
| DDE_FBUSY        | Application is busy  |
| DDE_FNODATA      | No data transfer for advise  |
| DDE_FACKREQ      | Acknowledgements are requested   |
| DDE_FRESPONSE    | Response to WM_DDE_REQUEST   |
| DDE_NOTPROCESSED | DDE message not understood   |
| DDE_FAPPSTATUS   | A 1-byte field of bits that are reserved for application-specific returns. |

### **usFormat** (USHORT)

Data format.

One of the DDE data formats.

|             |   |
|-------------|---|
| DDEFMT_TEXT | Text format.  |
| Other       | DDE format registered with the atom manager, using the system atom table. The predefined DDE formats are guaranteed not to conflict with the values returned by the atom manager. |

### **offszItemName** (USHORT)

Offset to item name.

This is the offset to the item name from the start of this structure. Item name is a null (0x00) terminated string. If no item name exists, there must be a single null (0x00) character in this position. (That is, *ItemName* is ALWAYS a null terminated string.)

### **offabData** (USHORT)

Offset to beginning of data.

This is the offset to the data, from the start of this structure. This field should be calculated regardless of the presence of data. If no data exists, **cbData** must be zero (0).

For compatibility reasons, this data should not contain embedded pointers. Offsets should be used instead.

---

## DESKTOP

Desktop background state structure.

### Syntax

```
typedef struct _DESKTOP {
    ULONG        cbSize;
    HBITMAP      hbm;
    LONG         x;
    LONG         y;
    ULONG        fl;
    LONG         lTileCount;
    CHAR         szFile[260];
} DESKTOP;

typedef DESKTOP *PDESKTOP;
```

### Fields

#### **cbSize** (ULONG)

Length of structure.

#### **hbm** (HBITMAP)

Bit-map handle of desktop background.

#### **x** (LONG)

X desktop coordinate of the origin of the bit map.

#### **y** (LONG)

Y desktop coordinate of the origin of the bit map.

#### **fl** (ULONG)

Desktop background state indicators or setting options.

- |                     |  |
|---------------------|--|
| <b>SDT_CENTER</b>   | The desktop background bit map is, or is to be, centered on the screen. If this option is specified, then the values of the x the y parameters are inapplicable.   |
| <b>SDT_DESTROY</b>  | Any existing desktop background bit map is to be destroyed. The setting of this option is not returned on the WinQueryDesktopBkgnd function.   |
| <b>SDT_LOADFILE</b> | For the WinSetDesktopBkgnd function the bit map is to be loaded from the filename specified. If the SDT_NOBKGND flag is also set then the bit map is loaded but the background is not set. Tiling and scaling may be performed at load time or later when setting the bit map. |
| <b>SDT_NOBKGND</b>  | There is no desktop background bit map, that is the desktop background is a solid color. For the WinQueryDesktopBkgnd function the existing background is to be left unmodified unless SDT_DESTROY is also specified.  |

|             |   |
|-------------|---|
| SDT_PATTERN | The bit map represents a fill pattern.  |
| SDT_RETAIN  | The <i>szFile</i> is, or is to be, remembered for use when the system is started.   |
| SDT_SCALE   | The bit map is, or is to be, scaled to fill the desktop. If this option is specified, then the values of the <i>x</i> and <i>y</i> parameters are inapplicable. |
| SDT_TILE    | The bit map is, or is to be, tiled to fill the desktop.   |

#### **ITileCount (LONG)**

Number of images of the bit map to be tiled.

The tile count is the number of images to be drawn in the vertical and horizontal direction when tiling the desktop background.

#### **szFile[260] (CHAR)**

Zero-terminated name of the file containing the bit map.

## **DEVOPENSTRUC**

Open-device data structure.

### **Syntax**

```
typedef struct _DEVOPENSTRUC {
    PSZ          pszLogAddress;
    PSZ          pszDriverName;
    PDRIVDATA    pdriv;
    PSZ          pszDataType;
    PSZ          pszComment;
    PSZ          pszQueueProcName;
    PSZ          pszQueueProcParams;
    PSZ          pszSpoolerParams;
    PSZ          pszNetworkParams;
} DEVOPENSTRUC;

typedef DEVOPENSTRUC *PDEVOPENSTRUC;
```

### **Fields**

#### **pszLogAddress (PSZ)**

Logical address.

This is required for an OD\_DIRECT device being opened with DevOpenDC; it is the logical device address, such as "LPT1" on OS/2. Some drivers may accept a file name for this parameter, or even a named pipe.

Where output is to be queued (for an OD\_QUEUED device), this is the name of the queue for the output device. The queue name can be a UNC name.

**Note:** This parameter can be a port name for a printer device context.



**pszDriverName (PSZ)**

Driver name.

Character string identifying the printer driver, for example, LASERJET. The *pszDriverName* field of the PRQINFO3 structure, associated with the required print queue, gives the driver and device name, separated by a period, for example LASERJET.HP LaserJet IIID It can contain only the name up to the period, for example LASERJET.

**pdriv (PDRIVDATA)**

Driver data.

Data that is to be passed directly to the PM device driver. Whether any of this is required depends upon the device driver.

For printer device context, this is a pointer to the job properties data.

**pszDataType (PSZ)**

Data type.

For a OD\_QUEUED or OD\_DIRECT device, this parameter defines the type of data that is to be queued as follows:

PM\_Q\_STD Standard format

PM\_Q\_RAW Raw format

Note that a device driver can define other data types.

For OD\_QUEUED or OD\_DIRECT device types, the default is supplied by the device driver if *pszDataType* is not specified. For any other device type, *pszDataType* is ignored.

**pszComment (PSZ)**

Comment.

Optional character string that the printer object displays to the user in a job settings notebook. It is recommended that the application include its own name in this comment string.

**Note:** The job title text is derived from the document name passed to DevEscape (DEVESC\_STARTDOC).

**pszQueueProcName (PSZ)**

Queue-processor name.

This is the name of the queue processor for queued output, and is usually the default.

**pszQueueProcParams (PSZ)**

Queue-processor parameters.

Queue processor parameters (optional). They can include information such as the number of copies you want to print and the size of the output area on the printed page.

The first parameter (*COP*) is used for all spool-file formats. The remaining parameters are valid for PM\_Q\_STD spool files only. Because PM\_Q\_STD data are used mainly for *graphic* data, these parameters are described in relation to the printing of picture files.

The PMPRINT/PMPLOT queue-processor parameters are separated by spaces and are:

**COP=*n***

The *COP* parameter specifies the number of copies of the spool file that you want printed. The value of *n* must be an integer in the range of 1 through 999.

The default is *COP=1*.

**ARE=C | *w,h,l,t***

The *ARE* parameter determines the size and position of the output area. This is the area of the physical page to which printing is restricted.

The default value of *ARE=C* means that the output area is the whole page. Note, however, that the printer cannot print outside its own device clip limits.

To size and position the output area at a specific point on the page, use *ARE=w,h,l,t*, where:

**w, h** are the width and height of the desired output area.

**l, t** are the offsets of the upper-left corner of the output area from the left (*l*) and from the top (*t*) of the maximum output area.

These four values must be given as percentages of the maximum output dimensions. The maximum output area is the area within the device clip limits.

**FIT=S | *l,t***

The *FIT* parameter determines which part of the picture is to be printed. You can request the whole of the picture, scaled to fit the output area; or you can position the picture (actual size) anywhere within the output area. This could mean that the picture is clipped at the boundaries of the output area.

The default value of *FIT=S* causes the output to be scaled until the larger of the height or width just fits within the defined output area. The aspect ratio of the picture is maintained.

To print the picture in actual size, use *FIT=l,t*, where *l,t* are the coordinates of the point in the picture that you want positioned at the center of the output area: *l* is measured from the left edge of the picture; and *t* is measured from the top edge. The coordinates must be given as percentages of the actual dimensions of the picture.

**XFM=0 | 1**

The *XFM* parameter enables you to override the picture-positioning and clipping instructions that are provided by the *ARE* and *FIT* parameters, including their defaults.

The default value of *XFM=1* allows the appearance of the output to be determined by the settings of the *ARE* and *FIT* parameters.

A value of *XFM=0* yields output as specified in the picture file. For example, applications that use many different forms can define different positions on each form for their output.

**COL=M | C**

The *COL* parameter enables you to specify color output if you have a color printer.

A value of *COL=M* creates monochrome output (black foreground with no background color). This is supported by all devices.

A value of *COL=C* creates color output. If you request color output on a monochrome device, the printer presentation driver tries to satisfy your request, which can cause problems because the only color available is black. For example, if the picture file specifies a red line on a blue background, both are drawn in black.

The default is *COL=M* when you are addressing a monochrome printer and *COL=C* when you are addressing a color printer.

**MAP=N | A**

The *MAP* parameter enables you to decide how the *neutral* colors (those that are not specified in the picture file) are printed.

The default value of *MAP=N* yields a *normal* representation of the screen picture on a printed page, which means that the page background is white and the foreground is black.

A value of *MAP=A* provides the reverse of the normal representation: the background is black and the foreground is white on the printed page.

**CDP=codepage**

The *CDP* parameter overrides the codepage to be used for PM\_Q\_RAW print jobs. The print queue driver uses DEVESC\_SETMODE to set the codepage, but not all printer drivers support this device escape.

**XLT=0 | 1**

The *XLT* parameter can eliminate the translation component when printing a metafile if *XLT=1*.

When the resolution of the device is higher than that of the world coordinate space, a small translation of world coordinate point (0,0) occurs on the device to preserve the accuracy of the mapping from world to device coordinate units. For example, (0,0) becomes (1,1) if there are 3 pels to every world coordinate.

Normally, this is not noticeable, but it can be a problem with some devices. For example, in order to draw a complete row of 80 characters using a device font, a device may require the text to start at device coordinate position zero. Starting at a position other than zero may cause one or more characters at the end of the row to be clipped. In such cases, elimination of the translation is important and can be accomplished by specifying *XLT=1*.

The default is *XLT=0*.

## pszSpoolerParams (PSZ)

Spooler parameters.

Spooler parameters (optional) are separated by spaces. They are used for scheduling print jobs and are as follows:

- The form names that identify the paper to be used, for example, *FORM=A4,A5,ENV*. The form names are optional; but if they are provided, the spooler is able to hold off printing the jobs until the required form is installed in the printer. If the form name is not provided, the spooler attempts to print the job. The printer driver recognizes that there is a forms problem and displays a **FORMS MISMATCH** message box.
- Priority of the print job, for example, *PTY=60*. The priority is specified as an integer in the range 1 through 99; 99 is the highest. The default priority value is 50. The application can use the spooler priority parameter to prioritize its own jobs; however, it is not good practice for an application always to use priority 99 in an attempt to get its jobs printed first.

## pszNetworkParams (PSZ)

Network parameters.

Optional parameter that can be used to specify network options; for example, *USER=JOESMITH*.

---

## DLGTEMPLATE

Dialog-template structure.

### Syntax

```
typedef struct _DLGTEMPLATE {
    USHORT      cbTemplate;
    USHORT      type;
    USHORT      codepage;
    USHORT      offadlgti;
    USHORT      fsTemplateStatus;
    USHORT      iItemFocus;
    USHORT      coffPresParams;
    DLGITITEM   adlgti[1];
} DLGTEMPLATE;

typedef DLGTEMPLATE *PDLGTEMPLATE;
```

### Fields

#### cbTemplate (USHORT)

Length of template.

#### type (USHORT)

Template format type.

**codepage** (USHORT)

Code page.

**offadlgti** (USHORT)

Offset to dialog items.

**fsTemplateStatus** (USHORT)

Template status.

**iltemFocus** (USHORT)

Index of item to receive focus initially.

**coffPresParams** (USHORT)

Count of presentation-parameter offsets.

**adlgti[1]** (DLGTITEM)

Start of dialog items.

---

## DLGTITEM

Dialog-item structure.

### Syntax

```
typedef struct _DLGTITEM {
    USHORT    fsItemStatus;
    USHORT    cChildren;
    USHORT    cchClassLen;
    USHORT    offClassName;
    USHORT    cchTextLen;
    USHORT    offText;
    ULONG     flStyle;
    SHORT     x;
    SHORT     y;
    SHORT     cx;
    SHORT     cy;
    USHORT    id;
    USHORT    offPresParams;
    USHORT    offCtlData;
} DLGTITEM;

typedef DLGTITEM *PDLGTITEM;
```

### Fields

**fsItemStatus** (USHORT)

Status.

**cChildren** (USHORT)

Count of children to this dialog item.

**cchClassLen** (USHORT)

Length of class name.

If zero, *offClassName* contains the hexadecimal equivalent of a preregistered class name.

**offClassName** (USHORT)

Offset to class name.

If *cchClassLen* is nonzero, this is the offset to a null-terminated ASCII string that contains the classname. If *cchClassLen* is zero, this is of the form 0xhhhh, where hhhh is the hexadecimal equivalent of the preregistered class name.

**cchTextLen** (USHORT)

Length of text.

**offText** (USHORT)

Offset to text.

**fiStyle** (ULONG)

Dialog item window style.

The high-order 16 bits are the standard WS\_\* style bits. The low-order 16 bits are available for class-specific use.

**x** (SHORT)

X-coordinate of origin of dialog-item window.

**y** (SHORT)

Y-coordinate of origin of dialog-item window.

**cx** (SHORT)

Dialog-item window width.

**cy** (SHORT)

Dialog-item window height.

**id** (USHORT)

Identity.

**offPresParams** (USHORT)

Reserved.

**offCtlData** (USHORT)

Offset to control data.

## DRAGIMAGE

Dragged-object-image structure which describes the images that are to be drawn under the direct-manipulation pointer for the duration of a drag operation.

### Syntax

```
typedef struct _DRAGIMAGE {
    USHORT      cb;
    USHORT      cptl;
    LHANDLE     hImage;
    SIZEL       sizlStretch;
    ULONG       fl;
    SHORT       cxOffset;
    SHORT       cyOffset;
} DRAGIMAGE;

typedef DRAGIMAGE *PDRAGIMAGE;
```

### Fields

#### cb (USHORT)

Size, in bytes, of the DRAGIMAGE structure.

#### cptl (USHORT)

The number of points in the point array if *fl* is specified as DRG\_POLYGON.

#### hImage (LHANDLE)

Handle representing the image to display.

The type is determined by *fl*.

#### sizlStretch (SIZEL)

Dimensions for stretching when *fl* is specified as DRG\_STRETCH.

#### fl (ULONG)

Flags.

DRG\_ICON                    *hImage* is an HPOINTER.

DRG\_BITMAP                *hImage* is an HBITMAP.

DRG\_POLYGON               *hImage* is a pointer to an array of points that will be connected with GpiPolyLine to form a polygon. The first point of the array should be (0,0), and the other points should be placed relative to this position.

DRG\_STRETCH               If DRG\_ICON or DRG\_BITMAP is specified, the image is expanded or compressed to the dimensions specified by *sizlStretch*.

DRG\_TRANSPARENT           If DRG\_ICON is specified, an outline of the icon is generated and displayed instead of the original icon.

**DRG\_CLOSED**

If **DRG\_POLYGON** is specified, a closed polygon is formed by moving the current position to the last point in the array before calling **GpiPolyLine**.

**cxOffset** (SHORT)

X-offset from the pointer hot spot to the origin of the image.

**cyOffset** (SHORT)

Y-offset from the pointer hot spot to the origin of the image.

---

## DRAGINFO

Drag-information structure.

### Syntax

```
typedef struct DRAGINFO {
    ULONG      cbDraginfo;
    USHORT     cbDragitem;
    USHORT     usOperation;
    HWND       hwndSource;
    SHORT      xDrop;
    SHORT      yDrop;
    USHORT     cditem;
    USHORT     usReserved;
} DRAGINFO;

typedef DRAGINFO *PDRAGINFO;
```

### Fields

**cbDraginfo** (ULONG)

Structure size, in bytes.

The size includes the array of **DRAGITEM** structures.

**cbDragitem** (USHORT)

Size, in bytes, of each **DRAGITEM** structure.



**usOperation** (USHORT)

Modified drag operations.

An application can define its own modified drag operations for use when simulating a drop. These operations must have a value greater than DO\_UNKNOWN. Possible values are described in the following list:

DO\_DEFAULT Execute the default drag operation. No modifier keys are pressed.  
DO\_COPY Execute a copy operation. The Ctrl key is pressed.  
DO\_LINK Execute a link operation. The Ctrl+Shift keys are pressed.  
DO\_MOVE Execute a move operation. The Shift key is pressed.  
DO\_UNKNOWN An undefined combination of modifier keys is pressed.

**hwndSource** (HWND)

Window handle of the source of the drag operation.

**xDrop** (SHORT)

X-coordinate of drop point expressed in desktop coordinates.

**yDrop** (SHORT)

Y-coordinate of drop point expressed in desktop coordinates.

**cditem** (USHORT)

Count of DRAGITEM structures.

**usReserved** (USHORT)

Reserved.

---

## DRAGITEM

Drag-object structure.

### Syntax

```
typedef struct _DRAGITEM {
    HWND      hwndItem;
    ULONG     ulItemID;
    HSTR      hstrType;
    HSTR      hstrRMF;
    HSTR      hstrContainerName;
    HSTR      hstrSourceName;
    HSTR      hstrTargetName;
    SHORT     cxOffset;
    SHORT     cyOffset;
    USHORT    fsControl;
    USHORT    fsSupportedOps;
} DRAGITEM;

typedef DRAGITEM *PDRAGITEM;
```

## Fields

### **hwndItem** (HWND)

Window handle of the source of the drag operation.

### **ulItemID** (ULONG)

Information used by the source to identify the object being dragged.

### **hstrType** (HSTR)

String handle of the object type.

The string handle must be created using the `DrgAddStrHandle` function. The string is of the form:

```
type[, type...]
```

The first type in the list must be the true type of the object. The following types are used by the OS/2\* shell:

|                           |                      |
|---------------------------|----------------------|
| <code>DRT_ASM</code>      | Assembler code       |
| <code>DRT_BASIC</code>    | BASIC code           |
| <code>DRT_BINDATA</code>  | Binary data          |
| <code>DRT_BITMAP</code>   | Bit map              |
| <code>DRT_C</code>        | C code               |
| <code>DRT_COBOL</code>    | COBOL code           |
| <code>DRT_DLL</code>      | Dynamic link library |
| <code>DRT_DOSCMD</code>   | DOS command file     |
| <code>DRT_EXE</code>      | Executable file      |
| <code>DRT_FONT</code>     | Font                 |
| <code>DRT_FORTRAN</code>  | FORTRAN code         |
| <code>DRT_ICON</code>     | Icon                 |
| <code>DRT_LIB</code>      | Library              |
| <code>DRT_METAFILE</code> | Metafile             |
| <code>DRT_OS2CMD</code>   | OS/2 command file    |
| <code>DRT_PASCAL</code>   | Pascal code          |
| <code>DRT_RESOURCE</code> | Resource file        |
| <code>DRT_TEXT</code>     | Text                 |
| <code>DRT_UNKNOWN</code>  | Unknown type.        |

### **hstrRMF** (HSTR)

String handle of the rendering mechanism and format.

The string handle must be created using the `DrgAddStrHandle` function. The string is of the form:

```
mechfmt[, mechfmt...]
```

where mechfmt can be in either of the following formats:

- <mechanism(1),format(1)>
- (mechanism(1)[, mechanism(n)...]) x (format(1)[,format(n)...])

The first mechanism/format pair must be the native rendering mechanism and format of the object.

Valid mechanisms are:

|               |  |
|---------------|--|
| "DRM_DDE"     | Dynamic data exchange                            |
| "DRM_OBJECT"  | Item being dragged is a workplace object.        |
| "DRM_OS2FILE" | OS/2 file  |
| "DRM_PRINT"   | Object can be printed using direct manipulation. |

Valid formats are:

|                    |                        |
|--------------------|------------------------|
| "DRF_BITMAP"       | OS/2 bit map           |
| "DRF_DIB"          | DIB                    |
| "DRF_DIF"          | DIF                    |
| "DRF_DSPBITMAP"    | Stream of bit-map bits |
| "DRF_METAFILE"     | Metafile               |
| "DRF_OEMTEXT"      | OEM text               |
| "DRF_OWNERDISPLAY" | Bit stream             |
| "DRF_PTRPICT"      | Printer picture        |
| "DRF_RTF"          | Rich text              |
| "DRF_SYLK"         | SYLK                   |
| "DRF_TEXT"         | Null-terminated string |
| "DRF_TIFF"         | TIFF                   |
| "DRF_UNKNOWN"      | Unknown format.        |

**hstrContainerName** (HSTR)

String handle of the name of the container holding the source object.

The string handle must be created using the DrgAddStrHandle function.

**hstrSourceName** (HSTR)

String handle of the name of the source object.

The string handle must be created using the DrgAddStrHandle function.

**hstrTargetName** (HSTR)

String handle of the suggested name of the object at the target.

It is the responsibility of the source of the drag operation to create this string handle before calling DrgDrag.

**cxOffset** (SHORT)

X-offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from *cxOffset* in the DRAGIMAGE structure by DrgDrag.

**cyOffset** (SHORT)

Y-offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from *cyOffset* in the DRAGIMAGE structure by DrgDrag.

**fsControl** (USHORT)

Source-object control flags.

|                    |   |
|--------------------|---|
| DC_OPEN            | Object is open  |
| DC_REF             | Reference to another object   |
| DC_GROUP           | Group of objects  |
| DC_CONTAINER       | Container of other objects  |
| DC_PREPARE         | Source requires a DM_RENDERPREPARE message before it establishes a data transfer conversation |
| DC_REMOVEABLEMEDIA | Object is on removable media, or object cannot be recovered after a move operation.           |

**fsSupportedOps** (USHORT)

Direct manipulation operations supported by the source object.

|             |                          |
|-------------|--------------------------|
| DO_COPYABLE | Source supports DO_COPY  |
| DO_LINKABLE | Source supports DO_LINK  |
| DO_MOVEABLE | Source supports DO_MOVE. |

---

**DRAGTRANSFER**

Drag-conversation structure.

**Syntax**

```
typedef struct _DRAGTRANSFER {
    ULONG          cb;
    HWND           hwndClient;
    PDRAGITEM      pditem;
    HSTR           hstrSelectedRMF;
    HSTR           hstrRenderToName;
    ULONG          ullTargetInfo;
    USHORT         usOperation;
    USHORT         fsReply;
} DRAGTRANSFER;

typedef DRAGTRANSFER *PDRAGTRANSFER;
```

## Fields

### **cb** (ULONG)

Size, in bytes, of the structure.

### **hwndClient** (HWND)

Handle of the client window.

This can be the target window or a window that represents an object in a container that was dropped on.

### **pditem** (PDRAGITEM)

Pointer to the DRAGITEM structure that is to be rendered.

This structure must exist within the DRAGINFO structure that was passed in the DM\_DROP message.

### **hstrSelectedRMF** (HSTR)

String handle for the selected rendering mechanism and format for the transfer operation.

This handle must be created using DrgAddStrHandle. The target is responsible for deleting this handle when the conversation is complete. The string is in the format: <MECHANISM,FORMAT>.

### **hstrRenderToName** (HSTR)

String handle representing the name where the source places, and the target finds, the data that is rendered.

The target is responsible for deleting this string handle when the conversation terminates. The contents of this field vary according to the rendering mechanism. See *hstrRMF* field in DRAGITEM.

|           |   |
|-----------|---|
| OS/2 File | The string handle represents the fully qualified name of the file where the rendering will be placed. |
| DDE       | This field is not used.   |
| Print     | This field is not used.   |

### **ulTargetInfo** (ULONG)

Reserved.

Reserved for use by the target. The target can use this field for information about the object and rendering operation.

### **usOperation** (USHORT)

The operation.

Values are:

|         |   |
|---------|---|
| DO_COPY | Execute a copy operation.                 |
| DO_LINK | Execute a link operation.                 |
| DO_MOVE | Execute a move operation.                 |
| OTHER   | Execute an application-defined operation. |

## **fsReply** (USHORT)

Reply flags.

Replay flags for the message. These flags can be set as follows:

**DMFL\_NATIVE RENDER** The source does not support rendering for this object. A source should not set this flag unless it provides sufficient information at the time of the drop for the target to perform the rendering operation. The target must send **DM\_ENDCONVERSATION** to the source after carrying out the rendering operation, or when it elects not to do a native rendering.

**DMFL\_RENDERRETRY** The source supports rendering for the object, but does not support the selected rendering mechanism and format. The target can try another mechanism and format by sending another **DM\_RENDER** message. If the target does not retry, it must send a **DM\_RENDERCOMPLETE** message to the source. This flag is set in conjunction with the **DMFL\_NATIVE RENDER** flag.

---

## **DRIVDATA**

Driver-data structure.

### **Syntax**

```
typedef struct _DRIVDATA {
    LONG      cb;
    LONG      IVersion;
    CHAR      szDeviceName[32];
    CHAR      abGeneralData[1];
} DRIVDATA;

typedef DRIVDATA *PDRIVDATA;
```

### **Fields**

#### **cb** (LONG)

Length.

The length of the structure.

#### **IVersion** (LONG)

Version.

The version number of the data. Version numbers are defined by particular PM device drivers.

**szDeviceName[32]** (CHAR)

Device name.

A string in a 32-byte field, identifying the particular device (model number, and so on). Again, valid values are defined by PM device drivers.

**abGeneralData[1]** (CHAR)

General data.

Data as defined by the Presentation Manager device driver.

The data type of this field is defined by the Presentation Manager device driver. It does not contain pointers, as these are not necessarily valid when passed to the device driver.

---

## ENTRYFDATA

Entry-field control data structure.

### Syntax

```
typedef struct _ENTRYFDATA {
    USHORT      cb;
    USHORT      cchEditLimit;
    USHORT      ichMinSel;
    USHORT      ichMaxSel;
} ENTRYFDATA;

typedef ENTRYFDATA *PENTRYFDATA;
```

### Fields

**cb** (USHORT)

Length of control data in bytes.

The length of the control data for an entry field control.

**cchEditLimit** (USHORT)

Edit limit.

This is the maximum number of characters that can be entered into the entry field control.

If the operator tries to enter more text into an entry field control than is specified by the text limit set by the EM\_SETTEXTLIMIT message, the entry field control indicates the error by sounding the alarm and does not accept the characters.

**ichMinSel** (USHORT)

Minimum selection.

**ichMaxSel** (USHORT)

Maximum selection.

The *ichMinSel* and *ichMaxSel* parameters identify the current selection within the entry field control. Characters within the text with byte offsets less than the *ichMaxSel* parameter and greater than or equal to the *ichMinSel* parameter are the current selection. The cursor is positioned immediately before the character identified by the *ichMaxSel* parameter.

If the *ichMinSel* parameter is equal to the *ichMaxSel* parameter, the current selection becomes the insertion point.

If the *ichMinSel* parameter is equal to 0 and the *ichMaxSel* is greater than or equal to text limit set by the EM\_SETTEXTLIMIT message, the entire text is selected.

---

**ERRORID**

Error identity.

**Syntax**

```
typedef ULONG ERRORID;
```

---

**ERRINFO**

Error-information structure.

**Syntax**

```
typedef struct _ERRINFO {
    ULONG      cbFixedErrInfo;
    ERRORID    idError;
    ULONG      cDetailLevel;
    ULONG      offaoffszMsg;
    ULONG      offBinaryData;
} ERRINFO;

typedef ERRINFO *PERRINFO;
```

**Fields****cbFixedErrInfo** (ULONG)

Length of fixed data to this structure.

**idError** (ERRORID)

Error identity.

This is identical to the value returned by WinGetLastError.



**cDetailLevel (ULONG)**

Number of levels of detail.

This is the number of entries in the array of words pointed to by the following field. One level of detail is provided.

**offaoffszMsg (ULONG)**

Offset to the array of message offsets.

This is an offset to an array of 16-bit offsets to null-terminated strings. Each string is a printable message that offers varying levels of information. The first level is the least amount of detail, and the remaining levels offer more and more detail.

The first level of detail is always an error message string, in the following format:

xxxxnnns

where   xxx    is the product identifier  
           nnnn  is the message number  
           s     is the message severity letter  
                   W = warning  
                   E = error  
                   S = severe error  
                   U = unrecoverable

**offBinaryData (ULONG)**

Offset to the binary data.

This can contain additional information relating to the error.

**ESCMODE**

Structure for setting printer mode. See DevEscape (DEVESC\_SETMODE).

**Syntax**

```
typedef struct _ESCMODE {
    ULONG      mode;
    BYTE      modedata[1];
} ESCMODE;

typedef ESCMODE *PESCMODE;
```

This data structure is a more-general version of the of the ESCSETMODE data structure.

## Fields

**mode** (ULONG)  
Mode.

**modedata[1]** (BYTE)  
Mode data.

---

## ESCSETMODE

Structure for setting printer mode. See DevEscape (DEVESC\_SETMODE).

## Syntax

```
typedef struct _ESCSETMODE {
    ULONG      mode;
    USHORT     codepage;
} ESCSETMODE;

typedef ESCSETMODE *PESCSETMODE;
```

This data structure is a specific-case version of the ESCMODE data structure, used to set the code page of a printer.

## Fields

**mode** (ULONG)  
Mode to be set.

0 Set mode to specified code page. Any font can be used.

**codepage** (USHORT)  
Code page.

If zero is specified for the code page, the printer is set to the hardware default.

## FACENAMEDESC

Face-name description structure. See `GpiQueryFaceString`.

### Syntax

```
typedef struct _FACENAMEDESC {
    USHORT      usSize;
    USHORT      usWeightClass;
    USHORT      usWidthClass;
    USHORT      usReserved;
    ULONG       flOptions;
} FACENAMEDESC;

typedef FACENAMEDESC *PFACENAMEDESC;
```

### Fields

#### **usSize** (USHORT)

Length of structure.

#### **usWeightClass** (USHORT)

Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

|                                  |  |
|----------------------------------|--|
| <code>FWEIGHT_DONT_CARE</code>   | Any font weight satisfies the request. |
| <code>FWEIGHT_ULTRA_LIGHT</code> | Ultra-light.                           |
| <code>FWEIGHT_EXTRA_LIGHT</code> | Extra-light.                           |
| <code>FWEIGHT_LIGHT</code>       | Light.                                 |
| <code>FWEIGHT_SEMI_LIGHT</code>  | Semi-light.                            |
| <code>FWEIGHT_NORMAL</code>      | Medium (normal) weight.                |
| <code>FWEIGHT_SEMI_BOLD</code>   | Semi-bold.                             |
| <code>FWEIGHT_BOLD</code>        | Bold.                                  |
| <code>FWEIGHT_EXTRA_BOLD</code>  | Extra-bold.                            |
| <code>FWEIGHT_ULTRA_BOLD</code>  | Ultra-bold.                            |

#### **usWidthClass** (USHORT)

Width class.

Indicates the relative aspect ratio of the characters of the font in relation to the normal aspect ratio for this type of font:

|                                     |                                       |
|-------------------------------------|---------------------------------------|
| <code>FWIDTH_DONT_CARE</code>       | Any font width satisfies the request. |
| <code>FWIDTH_ULTRA_CONDENSED</code> | Ultra-condensed (50% of normal).      |
| <code>FWIDTH_EXTRA_CONDENSED</code> | Extra-condensed (62.5% of normal).    |
| <code>FWIDTH_CONDENSED</code>       | Condensed (75% of normal).            |
| <code>FWIDTH_SEMI_CONDENSED</code>  | Semi-condensed (87.5% of normal).     |
| <code>FWIDTH_NORMAL</code>          | Medium (normal).                      |
| <code>FWIDTH_SEMI_EXPANDED</code>   | Semi-expanded (112.5% of normal).     |
| <code>FWIDTH_EXPANDED</code>        | Expanded (125% of normal).            |

FWIDTH\_EXTRA\_EXPANDED Extra-expanded (150% of normal).  
FWIDTH\_ULTRA\_EXPANDED Ultra-expanded (200% of normal).

**usReserved** (USHORT)  
Reserved.

**flOptions** (ULONG)  
Other characteristics of the font.

FTYPE\_ITALIC Italic font required. If not specified, non-italic font required.

FTYPE\_ITALIC\_DONT\_CARE Italic and non-italic fonts can satisfy the request. If this option is specified, FTYPE\_ITALIC is ignored.

FTYPE\_OBLIQUE Oblique font required. If not specified, non-oblique font required.

FTYPE\_OBLIQUE\_DONT\_CARE Oblique and non-oblique fonts can satisfy the request. If this option is specified, FTYPE\_OBLIQUE is ignored.

FTYPE\_ROUNDED Rounded font required. If not specified, non-rounded font required.

FTYPE\_ROUNDED\_DONT\_CARE Rounded and non-rounded fonts can satisfy the request. If this option is specified, FTYPE\_ROUNDED is ignored.

---

## FATTRS

Font-attributes structure.

### Syntax

```
typedef struct _FATTRS {
    USHORT    usRecordLength;
    USHORT    fsSelection;
    LONG      lMatch;
    CHAR      szFacename[FACE_SIZE];
    USHORT    idRegistry;
    USHORT    usCodePage;
    LONG      lMaxBaselineExt;
    LONG      lAveCharWidth;
    USHORT    fsType;
    USHORT    fsFontUse;
} FATTRS;

typedef FATTRS *PFATTRS;
```

## Fields

### **usRecordLength** (USHORT)

Length of record.

### **fsSelection** (USHORT)

Selection indicators.

Flags causing the following features to be simulated by the system.

**Note:** If an italic flag is applied to a font that is itself defined as italic, the font is slanted further by italic simulation.

Underscore or strikeout lines are drawn using the appropriate attributes (for example, color) from the character bundle (see the CHARBUNDLE datatype), not the line bundle (see LINEBUNDLE). The width of the line, and the vertical position of the line in font space, are determined by the font. Horizontally, the line starts from a point in font space directly above or below the start point of each character, and extends to a point directly above or below the escapement point for that character.

For this purpose, the start and escapement points are those applicable to left-to-right or right-to-left character directions (see GpiSetCharDirection in *Graphics Programming Interface Programming Reference*), even if the string is currently being drawn in a top-to-bottom or bottom-to-top direction.

For left-to-right or right-to-left directions, any white space generated by the character extra and character break extra attributes (see GpiSetCharExtra and GpiSetCharBreakExtra in *Graphics Programming Interface Programming Reference*), as well as increments provided by the vector of increments on GpiCharStringPos and GpiCharStringPosAt, are also underlined/overstruck, so that in these cases the line is continuous for the string.

|                      |   |
|----------------------|---|
| FATTR_SEL_ITALIC     | Generate <i>italic</i> font.  |
| FATTR_SEL_UNDERSCORE | Generate <u>underscored</u> font.   |
| FATTR_SEL_BOLD       | Generate <b>bold</b> font. (Note that the resulting characters are wider than those in the original font.)  |
| FATTR_SEL_STRIKEOUT  | Generate font with <del>overstruck</del> characters.  |
| FATTR_SEL_OUTLINE    | Use an outline font with hollow characters. If this flag is not set, outline font characters are filled. Setting this flag normally gives better performance, and for sufficiently small characters (depending on device resolution) there may be little visual difference. |

### **IMatch** (LONG)

Matched-font identity.

### **szFacename[FACESIZE]** (CHAR)

Typeface name.

The typeface name of the font, for example, Tms Rmn.

**idRegistry** (USHORT)

Registry identifier.

Font registry identifier (zero if unknown).

**usCodePage** (USHORT)

Code page.

If zero, the current Gpi code page (see GpiSetCp in *Graphics Programming Interface Programming Reference*) is used. A subsequent GpiSetCp function changes the code page used for this logical font.

**lMaxBaselineExt** (LONG)

Maximum baseline extension.

For raster fonts, this should be the height of the required font, in world coordinates.

For outline fonts, this should be zero.

**lAveCharWidth** (LONG)

Average character width.

For raster fonts, this should be the width of the required font, in world coordinates.

For outline fonts, this should be zero.

**fsType** (USHORT)

Type indicators.

|                        |  |
|------------------------|--|
| FATTR_TYPE_KERNING     | Enable kerning (PostScript** only).                                      |
| FATTR_TYPE_MBCS        | Font for mixed single- and double-byte code pages.                       |
| FATTR_TYPE_DBCS        | Font for double-byte code pages.   |
| FATTR_TYPE_ANTIALIASED | Antialiased font required. Only valid if supported by the device driver. |

**fsFontUse** (USHORT)

Font-use indicators.

These flags indicate how the font is to be used. They affect presentation speed and font quality.

|                             |  |
|-----------------------------|--|
| FATTR_FONTUSE_NOMIX         | Text is not mixed with graphics and can be written without regard to any interaction with graphics objects.  |
| FATTR_FONTUSE_OUTLINE       | Select an outline (vector) font. The font characters can be used as part of a path definition. If this flag is not set, an outline font might or might not be selected. If an outline font is selected, however, character widths are rounded to an integral number of pels. |
| FATTR_FONTUSE_TRANSFORMABLE | Characters can be transformed (for example, scaled, rotated, or sheared).  |

---

## FFDESCS

Font-file descriptor.

### Syntax

```
typedef CHAR FFDESCS[2][FACESIZE];
```

---

## FIELDINFO

Structure that contains information about column data in the details view of the container control. The details view displays each FIELDINFO structure as a column of data that contains specific information about each container record. For example, one FIELDINFO structure, or column, might contain icons or bit maps that represent each container record. Another FIELDINFO structure might contain the date or time that each container record was created.

### Syntax

```
typedef struct _FIELDINFO {
    ULONG          cb;
    ULONG          flData;
    ULONG          flTitle;
    PVOID          pTitleData;
    ULONG          offStruct;
    PVOID          pUserData;
    struct _FIELDINFO *pNextFieldInfo;
    ULONG          cxWidth;
} FIELDINFO;

typedef FIELDINFO *PFIELDINFO;
```

### Fields

**cb** (ULONG)

Structure size.

The size (in bytes) of the FIELDINFO structure.

## **fiData (ULONG)**

Data attributes.

Attributes of the data in a field.

- Specify one of the following for each column to choose the type of data that is displayed in each column:

### **CFA\_BITMAPORICON**

The column contains bit-map or icon data.

### **CFA\_DATE**

The data in the column is displayed in date format. National Language Support (NLS) is enabled for date format. Use the data structure described in CDATE

### **CFA\_STRING**

Character or text data is displayed in this column.

### **CFA\_TIME**

The data in the column is displayed in time format. National Language Support (NLS) is enabled for time format. Use the data structure described in CTIME.

### **CFA\_ULONG**

Unsigned number data is displayed in this column. National Language Support (NLS) is enabled for number format.

- Specify any or all of the following column attributes:

### **CFA\_FIREADONLY**

Prevents text in a FIELDINFO data structure (text in a column) from being edited directly. This attribute applies only to columns for which the CFA\_STRING attribute has been specified.

### **CFA\_HORZSEPARATOR**

A horizontal separator is provided beneath column headings.

### **CFA\_INVISIBLE**

Invisible container column. The default is visible.

### **CFA\_OWNER**

Ownerdraw is enabled for this container column.

### **CFA\_SEPARATOR**

A vertical separator is drawn after this column.

- Specify one of the following for each column to vertically position data in that column:

### **CFA\_BOTTOM**

Bottom-justifies field data.

### **CFA\_TOP**

Top-justifies field data.

### **CFA\_VCENTER**

Vertically centers field data. This is the default.



- Specify one of the following for each column to horizontally position data in that column. These attributes can be combined with the attributes used for vertical positioning of column data by using an OR operator (|).

**CFA\_CENTER**

Horizontally centers field data.

**CFA\_LEFT**

Left-justifies field data. This is the default.

**CFA\_RIGHT**

Right-justifies field data.

**fiTitle (ULONG)**

Column heading attributes.

- Specify the following if icon or bit-map data is to be displayed in the column heading:

**CFA\_BITMAPORICON**

The column heading contains icon or bit-map data. If CFA\_BITMAPORICON is not specified, any data that is assigned to a column heading is assumed to be character or text data.

- Specify the following to prevent direct editing of a column heading:

**CFA\_FITTLEReadONLY**

Prevents a column heading from being edited directly.

- Specify one of the following for each column heading to vertically position data in that column heading:

**CFA\_TOP**

Top-justifies column headings.

**CFA\_BOTTOM**

Bottom-justifies column headings.

**CFA\_VCENTER**

Vertically centers column headings. This is the default.

- Specify one of the following for each column heading to horizontally position data in that column heading. These attributes can be combined with the attributes used for vertical positioning of column heading data by using an OR operator (|).

**CFA\_CENTER**

Horizontally centers column headings.

**CFA\_LEFT**

Left-justifies column headings. This is the default.

**CFA\_RIGHT**

Right-justifies column headings.

**pTitleData (PVOID)**

Column heading data.

Column heading data, which can be a text string, or an icon or bit map. The default is a

text string. If the *fTitle* field is set to the CFA\_BITMAPORICON attribute, this must be an icon or bit map.

**offStruct (ULONG)**

Structure offset.

Offset from the beginning of a RECORDCORE structure to the data that is displayed in this column.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

**pUserData (PVOID)**

Pointer to user data.

**pNextFieldInfo (struct \_FIELDINFO \*)**

Pointer to the next linked FIELDINFO data structure.

**cxWidth (ULONG)**

Column width.

Used to specify the width of a column. The default is an automatically sized column that is always the width of its widest element. If this field is set and the data is too wide, the data is truncated.

---

## FIELDINFOINSERT

Structure that contains information about the FIELDINFO structure or structures that are being inserted into a container. This structure is used in the CM\_INSERTDETAILFIELDINFO container message only. See "CM\_INSERTDETAILFIELDINFO" on page 22-44 for information about that message.

### Syntax

```
typedef struct _FIELDINFOINSERT {
    ULONG          cb;
    PFIELDINFO     pFieldInfoOrder;
    ULONG          fInvalidateFieldInfo;
    ULONG          cFieldInfoInsert;
} FIELDINFOINSERT;

typedef FIELDINFOINSERT *PFIELDINFOINSERT;
```

### Fields

**cb (ULONG)**

Structure size.

The size (in bytes) of the FIELDINFOINSERT structure.

**pFieldInfoOrder** (PFIELDINFO)

Column order.

Orders the FIELDINFO structure or structures relative to other FIELDINFO structures in the container. The values can be:

- CMA\_FIRST Places a FIELDINFO structure, or list of FIELDINFO structures, at the front of the list of columns.
- CMA\_END Places a FIELDINFO structure, or list of FIELDINFO structures, at the end of the list of columns.
- Other Pointer to a FIELDINFO structure that this structure, or list of structures, is to be inserted after.

**fInvalidateFieldInfo** (ULONG)

Update flag.

Flag that indicates an automatic display update after the FIELDINFO structures are inserted.

- TRUE The display is automatically updated after FIELDINFO structures are inserted.
- FALSE The application must send the CM\_INVALIDATEDDETAILFIELDINFO message after the FIELDINFO structures are inserted.

**cFieldInfoInsert** (ULONG)

Number of columns.

The number of FIELDINFO structures to be inserted. The *cFieldInfoInsert* field value must be greater than 0.

## FILEDLG

File-dialog structure.

### Syntax

```
typedef struct FILEDLG {
    ULONG      cbSize;
    ULONG      fl;
    ULONG      ulUser;
    LONG       lReturn;
    LONG       lSRC;
    PSZ        pszTitle;
    PSZ        pszOKButton;
    PFNWP      pfnDlgProc;
    PSZ        pszIType;
    PAPSZ      papszITypeList;
    PSZ        pszIDrive;
    PAPSZ      papszIDriveList;
    HMODULE    hMod;
    CHAR       szFullFile[CCHMAXPATH];
    PAPSZ      papszFQfilename;
    ULONG      ulFQFcount;
    USHORT     usDlgID;
    SHORT      x;
    SHORT      y;
    SHORT      sEAType;
} FILEDLG;

typedef FILEDLG *PFILEDLG;
```

### Fields

#### **cbSize** (ULONG)

Structure size.

Size of the structure. This field allows future expansion of the structure and must be initialized with the size of the FILEDLG structure.

## fl (ULONG)

FDS\_\* flags.

Several flags can be specified to alter the behavior of the dialog.

**Note:** The dialog must be either an “Open” or a “Save As” dialog. If neither the FDS\_OPEN\_DIALOG nor the FDS\_SAVEAS\_DIALOG flag is set, or if both are set, the dialog will return an error.

|                     |   |
|---------------------|---|
| FDS_APPLYBUTTON     | An Apply push button is added to the dialog. This is useful in a modeless dialog.   |
| FDS_CENTER          | The dialog is positioned in the center of its parent window, overriding any specified x, y position.  |
| FDS_CUSTOM          | A custom dialog template is used to create the dialog. The <i>hMod</i> and <i>usDlgID</i> fields must be initialized.   |
| FDS_ENABLEFILELB    | When this flag is set, the Files list box on a Save As dialog is enabled. When this flag is not set, the Files list box is not enabled for a Save As dialog. This is the default.   |
| FDS_FILTERUNION     | When this flag is set, the dialog uses the union of the string filter and the extended-attribute type filter when filtering files for the Files list box. When this flag is not set, the list box, by default, uses the intersection of the two.  |
| FDS_HELPBUTTON      | A Help push button of style (BS_HELP BS_NOPOINTINTERFOCUS) with an ID of DID_HELP_PB is added to the dialog. When this push button is pressed, a WM_HELP message is sent to <i>hwndO</i> .  |
| FDS_INCLUDE_EAS     | If this flag is set, the dialog will always query extended attribute information for files as it fills the Files list box. The default is to not query the information unless an extended attribute type filter has been selected.  |
| FDS_MODELESS        | When this flag is set, the dialog is modeless; WinFileDialog returns immediately after creating the dialog window and returns the window handle to the application. The application should treat the dialog as if it were created with WinLoadDlg. As in the modal (default) dialog case, the return value is found in the <i>IReturn</i> field of the FILEDLG structure passed to WinFileDialog. |
| FDS_MULTIPLESEL     | When this flag is set, the Files list box for the dialog is a multiple selection list box. When this flag is not set, the default is a single-selection list box.   |
| FDS_OPEN_DIALOG     | The dialog is an “Open” dialog when this flag is set.   |
| FDS_PRELOAD_VOLINFO | If this flag is set, the dialog will preload the volume information for the drives and will preset the current default directory for each drive. The default behavior is for the volume label to be blank and the initial directory will be the root directory for each drive.  |
| FDS_SAVEAS_DIALOG   | The dialog is a “Save As” dialog when this flag is set.   |

**ulUser (ULONG)**

Used by the application.

This field can be used by an application that is subclassing the file dialog to store its own state information.

**IReturn (LONG)**

Result code.

Result code from dialog dismissal. This field contains the ID of the push button pressed to dismiss the dialog, `DID_OK` or `DID_CANCEL`, unless the application supplies additional push buttons in its template. If an error occurs on dialog invocation, this field is set to zero.

**ISRC (LONG)**

System return code.

This field contains an `FDS_ERR` return code. When a dialog fails, this field is used to tell the application the reason for the failure.

**pszTitle (PSZ)**

Dialog title string.

When this field is `NULL`, the dialog title defaults to the name of the dialog currently running.

**pszOKButton (PSZ)**

OK push button text.

This string is used to set the text of the OK push button. The default text is `OK`.

**pfnDlgProc (PFNWP)**

Custom dialog procedure.

`NULL` unless the caller is subclassing the file dialog. When non-`NULL`, it points to the dialog procedure of the application.

**pszIType (PSZ)**

Extended-attribute type filter.

This field contains a pointer to the initial extended-attribute type filter that is applied to the initial dialog screen. This filter is not required to be in *papszITypeList*.

**papszITypeList (PAPSZ)**

Pointer to a table of pointers to extended-attribute types.

Each pointer in the table points to a null-terminated string, and each string is an extended-attribute type. These types are sorted in ascending order in the Type drop-down box. The end of the table is marked by a null pointer. To specify an empty table, the application sets this field to `NULL`, or it specifies a table containing only a null pointer.

**pszIDrive (PSZ)**

The initial drive.

This field contains a pointer to a string that specifies the initial drive applied to the initial dialog screen. This drive is not required to be in *papszDriveList*.

**papszDriveList (PAPSZ)**

Pointer to a table of pointers to drives.

Each pointer in the table points to a null-terminated string, and each string is a valid drive or network identifier. These drives and network IDs will be sorted in ascending order in the Drive drop-down box. The end of the table is marked by a null pointer. To specify an empty table, the application sets this field to NULL, or it specifies a table containing only a null pointer.

**hMod (HMODULE)**

Module for custom dialog resources.

If FDS\_CUSTOM is set, this is the HMODULE from which the custom file dialog template is loaded. NULLHANDLE causes the dialog resource to be pulled from the module of the current EXE.

**szFullFile[CCHMAXPATH] (CHAR)**

Character array.

An array of characters where CCHMAXPATH is a system-defined constant. On initialization, this field contains the initial fully-qualified path and file name. On completion, this field contains the selected fully-qualified path and file name. The simple file name can be replaced with a string filter, such as \*.DAT. When the dialog is invoked, all drive and path information is stripped from the entry and moved to the corresponding fields in the dialog.

When a file name is specified, the Files list box is scrolled to the matching file name. When there is no exact match, the closest match is used.

When a string filter is specified, the dialog is initially refreshed using the results of this filter intersected with the results of *pszIType*. After the dialog is initially shown, the string filter remains in the file name field until a file is selected, or the user overtypes the value.

When a file is selected, **szFullFile** is returned to the calling application and is set to the selected fully-qualified file name.

When more than one file is selected in a multiple file selection dialog, only the topmost selected file name is returned in this field.

**papszFQFilename (PAPSZ)**

Pointer to a table of pointers to fully-qualified file names.

Returned to multiple file selection dialogs when the user selects one or more files from the list box. If the user types the file name in the file name entry field, the file name will be in **szFullFile** and this pointer will be NULL. When one or more selections are made, the count of items in this array will be returned in *ulFQFCount*.

This table of pointers is storage allocated by the file dialog. When the application completes opening or saving all of the files specified, the application must call `WinFreeFileDlgList` to free the storage allocated by the file dialog.

**ulFQFCount** (ULONG)

Number of file names.

Number of file names selected in the dialog. In a single file selection dialog, this value is 1. In a multiple file selection dialog, this value will be the number of files selected by the user.

**usDlgID** (USHORT)

Custom dialog ID.

The ID of the dialog window. When `FDS_CUSTOM` is set, this field contains the ID of the resource containing the custom dialog template.

**x** (SHORT)

X-axis dialog position.

This, along with *y* and *hwndP*, is used to position the dialog. It is updated in the structure if the user moves the dialog to a new position. If the `FILEDLG` structure is reused, the dialog appears in the position at which it was left each time it is invoked. The `FDS_CENTER` flag overrides this position and automatically centers the dialog in its parent.

**y** (SHORT)

Y-axis dialog position.

This, along with *x* and *hwndP*, is used to position the dialog. It is updated in the structure if the user moves the dialog to a new position. If the `FILEDLG` structure is reused, the dialog appears in the position at which it was left each time it is invoked. The `FDS_CENTER` flag overrides this position and automatically centers the dialog in its parent.

**sEAType** (SHORT)

Selected extended-attribute type.

Returns a selected extended-attribute type to assign to the file name returned in **szFullFile**. This field is a zero-based offset into the *papszITypeList* and is returned only when the Save As dialog is used. A -1 value is returned when the Open dialog is used.

---

**FIXED**

Signed-integer fraction (16:16). This can be treated as a LONG where the value has been multiplied by 65 536.

**Syntax**

```
typedef LONG FIXED;
```



## FONTDLG

Font-dialog structure.

### Syntax

```
typedef struct _FONTDLG {
    ULONG        cbSize;
    HPS          hpsScreen;
    HPS          hpsPrinter;
    PSZ          pszTitle;
    PSZ          pszPreview;
    PSZ          pszPtSizeList;
    PFNWP        pfnDlgProc;
    PSZ          pszFamilyname;
    FIXED        fxPointSize;
    ULONG        fl;
    ULONG        flFlags;
    ULONG        flType;
    ULONG        flTypeMask;
    ULONG        flStyle;
    ULONG        flStyleMask;
    LONG         clrFore;
    LONG         clrBack;
    ULONG        ulUser;
    LONG         lReturn;
    LONG         lSRC;
    LONG         lEmHeight;
    LONG         lXHeight;
    LONG         lExternalLeading;
    HMODULE      hMod;
    FATTRS       fAttr;
    SHORT        sNominalPointSize;
    USHORT       usWeight;
    USHORT       usWidth;
    SHORT        x;
    SHORT        y;
    USHORT       usDlgId;
    USHORT       usFamilyBufLen;
    USHORT       usReserved;
} FONTDLG;

typedef FONTDLG *PFONTDLG;
```

### Fields

#### cbSize (ULONG)

Structure size.

This field allows for future expansion of the structure, and must be initialized with the size of the FONTDLG structure.

**hpsScreen** (HPS)

Screen presentation space.

If not NULLHANDLE, the screen presentation space from which screen fonts are queried.

**hpsPrinter** (HPS)

Printer presentation space.

If not NULLHANDLE, the printer presentation space from which printer font are queried.

**pszTitle** (PSZ)

Dialog title string.

Application-provided dialog title. If NULL, it defaults to "Font."

**pszPreview** (PSZ)

Font-preview window string.

String to show in font-preview window. If NULL, it defaults to "abcdABCD."

**Note:** Care is necessary when choosing the string to put in this field. Using many different characters causes excess memory to be used by the font cache.

**pszPtSizeList** (PSZ)

Application-provided point size list.

String which contains a list of point sizes to be used as the default list for outline fonts in the point-size drop-down area. Point sizes are separated by spaces. If NULL, the point size drop down defaults to 8, 10, 12, 14, 18, and 24.

**pfnDlgProc** (PFNWP)

Custom dialog procedure.

NULL unless the caller is subclassing the font dialog. When non-NULL, it points to the dialog procedure of the application.

**pszFamilyname** (PSZ)

Family name buffer.

Buffer provided by the application for passing the family name of the font. The font family name used by the application to select a font. When the first character in this string is NULL, no family name was initially selected, and the dialog defaults to the system font.

A buffer must be passed to the font dialog to allow the dialog to return the selected font family name. The size of this buffer is placed in the *usFamilyBufLen* field.

**fxPointSize** (FIXED)

Point size of the font.

If FNTS\_OWNERDRAWPREVIEW is set, 0 means the user wants to leave the font size unchanged and the application must update the preview area.

**fl** (ULONG)

FNTS\_\* flags.

|                         |   |
|-------------------------|---|
| FNTS_APPLYBUTTON        | An Apply push button is added to the dialog. This is useful in a modeless dialog.   |
| FNTS_BITMAPONLY         | The dialog presents bit-map fonts only. An application that changes fonts by using the presentation parameters (PP_* values) could use this flag.   |
| FNTS_CENTER             | The dialog is positioned in the center of its parent window, overriding any specified x,y position.   |
| FNTS_CUSTOM             | A custom dialog template is used to create the dialog. The <i>hMod</i> and <i>usDlgId</i> fields must be initialized.   |
| FNTS_FIXEDWIDTHONLY     | The dialog presents fixed-width (monospace) fonts only.   |
| FNTS_HELPBUTTON         | A Help push button of style (BS_HELP BS_NOPOINTERFOCUS) with an ID of DID_HELP_BUTTON is added to the dialog. If the push button is pressed, a WM_HELP message is sent to the <i>hwndO</i> parameter of the WinFontDlg function call.   |
| FNTS_INITFROMFATTRS     | The dialog initializes itself from the font attribute structure (FATTRS) that is passed.  |
| FNTS_MODELESS           | The dialog is modeless; WinFontDlg returns immediately after creating the dialog window and returns the window handle to the application. The application should treat the dialog as if it were created with WinLoadDlg. As in the modal (default) dialog case, the return value is found in the <i>IReturn</i> field of the FONDLG structure passed to WinFontDlg. |
| FNTS_NOSYNTHESIZEDFONTS | The dialog does not synthesize any fonts.   |
| FNTS_OWNERDRAWPREVIEW   | This flag makes the check boxes in the font dialog three-state check boxes, enabling the user to leave certain style attributes unchanged. Additionally, a WM_DRAWITEM message will be sent to the owner, providing the owner an opportunity to draw the preview window itself.   |
| FNTS_PROPORTIONALONLY   | The dialog presents proportionally spaced fonts only.   |
| FNTS_RESETBUTTON        | A Reset push button is added to the dialog. When this push button is pressed, the values for the dialog are restored to their initial values.   |
| FNTS_VECTORONLY         | The dialog presents vector fonts only.  |
| <b>flFlags (ULONG)</b>  |   |
| FNTF_* flags.           |   |
| FNTF_NOVIEWPRINTERFONTS | This flag is initialized only when both <i>hpsScreen</i> and <i>hpsPrinter</i> are not NULLHANDLE. On input, this parameter determines whether the printer fonts  |

are to be included in the font list box. The user controls this with a check box.

**FNTF\_NOVIEWSCREENFONTS**

This flag is initialized only when both *hpsScreen* and *hpsPrinter* are not NULLHANDLE. On input, this parameter determines whether the screen fonts should be included in the font list box. The user controls this with a check box.

**FNTF\_PRINTERFONTSELECTED**

This determines if a printer-specific font is selected by the user. The application should make an approximation of this printer font when outputting to the screen. This is an output-only flag and is ignored on input.

**FNTF\_SCREENFONTSELECTED**

This determines if a screen-specific font is selected by the user. The application should make an approximation of this screen font when outputting to the screen. This is an output-only flag and is ignored on input.

**flType (ULONG)**

The selected type bits.

These flags specify what additional attributes the user specified for the font. This field is used as the *flOptions* field in the FACENAMEDESC structure for GpiQueryFaceString.

**flTypeMask (ULONG)**

Mask of type bits to use.

This field is used only if FNTS\_OWNERDRAWPREVIEW is specified. It tells which flags of the *flTypeMask* field the user wants to change, and is relevant only if the text for which the font is selected has different faces and styles.

**flStyle (ULONG)**

Selected style bits.

Flags for any additional selections the user specified for the font. This field is used as the *fsSelection* field in the FATTRS structure passed to GpiCreateLogFont.

**flStyleMask (ULONG)**

Mask of style bits to use.

This field is used only if FNTS\_OWNERDRAWPREVIEW is specified. It tells which flags of the *flStyle* field the user wants to change and is relevant only if the text for which the font is selected has different faces and styles.

**clrFore (LONG)**

Font foreground color.

Foreground color of the font. This color is a value used for the color mode that *hpsScreen* is in. If FNTS\_OWNERDRAWPREVIEW is specified, this value can be CLR\_NOINDEX, leaving the foreground color "as is."

**clrBack** (LONG)

Font background color.

Background color of the font. This color is a value used for the color mode that *hpsScreen* is in. If `FNTS_OWNERDRAWPREVIEW` is specified, this value can be `CLR_NOINDEX` leaving the background color “as is.”

**ulUser** (ULONG)

Application-defined.

A ULONG that an application uses to store its state information when it is subclassing the font dialog.

**IReturn** (LONG)

Return value.

Return value from `WinFontDlg`. This value is the ID of the push button pressed to dismiss the dialog, `DID_OK` or `DID_CANCEL`, unless the application supplied additional push buttons in its template.

**ISRC** (LONG)

System return code.

This field contains an `FNTS_ERR` return code. When a dialog fails, this field is used to tell the application the reason for the failure.

**IEmHeight** (LONG)

Em height.

The Em height of the current font. This is the same as in the `FONTMETRICS` structure. It is an output-only parameter and its value has no effect on the behavior of the font dialog, but is updated when the user dismisses the dialog.

**IXHeight** (LONG)

X height.

The x height of the current font. This is the same as in the `FONTMETRICS` structure. It is an output-only parameter and its value has no effect on the behavior of the font dialog, but is updated when the user dismisses the dialog.

**IExternalLeading** (LONG)

External leading.

The external leading of the font. This is the same as in the `FONTMETRICS` structure. It is an output-only parameter and its value has no effect on the behavior of the font dialog, but is updated when the user dismisses the dialog.

**hMod** (HMODULE)

Module for custom dialog resources.

If `FNTS_CUSTOM` is set, this is the `HMODULE` from which the custom font dialog template is loaded. `NULLHANDLE` causes the dialog resource to be pulled from the module of the current EXE.

**fAttrs** (FATTRS)

Font-attribute structure.

Font-attribute structure of selected font. The FATTRS for the selected font. This is output-only for all fields except *usCodePage*, which is input/output, and the initial code page value passed is used for font selection. The value returned is the one for the matching font.

**sNominalPointSize** (SHORT)

Font point size.

The nominal point size of the font. This is the same as in the FONTMETRICS structure. It is an output-only parameter and its value has no effect on the behavior of the font dialog, but is updated when the user dismisses the dialog.

**usWeight** (USHORT)

Font weight.

The weight of the font. This is the weight-class/boldness the user selects for the font. This field is used as the *usWeightClass* field in the FACENAMEDESC structure for GpiQueryFaceString. When FNTS\_OWNERDRAWPREVIEW is set, 0 causes the application to leave the font weight "as is" and the application must update the preview area.

**usWidth** (USHORT)

Font width.

The width of the font. This is the width-class the user selects for the font. This field is used as the *usWidthClass* field in the FACENAMEDESC structure for GpiQueryFaceString. When FNTS\_OWNERDRAWPREVIEW is set, 0 causes the application to leave the font width "as is" and the application must update the preview area.

**x** (SHORT)

The x-axis dialog position.

This, along with *y* and *hwndP*, is used to position the dialog. It is updated in the structure if the user moves the dialog to a new position. This way, the dialog appears in the position at which it was left each time it is invoked. The FNTS\_CENTER flag overrides this position and automatically centers the dialog in its parent.

**y** (SHORT)

The y-axis dialog position.

This, along with *x* and *hwndP*, is used to position the dialog. It is updated in the structure if the user moves the dialog to a new position. This way, the dialog appears in the position at which it was left each time it is invoked. The FNTS\_CENTER flag overrides this position and automatically centers the dialog in its parent.

**usDlgId** (USHORT)

Dialog ID.

This sets the ID of the dialog window. If FNTS\_CUSTOM is set, this is the ID of the resource that contains the custom dialog template.

**usFamilyBufLen** (USHORT)

Buffersize.

Size of the buffer passed in the *pszFamilyname* resource that contains the custom dialog template.

**usReserved** (USHORT)

Reserved.

This is a reserved field.

---

## FONTMETRICS

Font-metrics structure.

This structure is returned to applications on the `GpiQueryFonts` and `GpiQueryFontMetrics` calls and conveys information from the font creator to the application.

### Syntax

```

typedef struct FONTMETRICS {
CHAR      szFamilyName[FACESIZE];
CHAR      szFaceName[FACESIZE];
USHORT    idRegistry;
USHORT    usCodePage;
LONG      lEmHeight;
LONG      lXHeight;
LONG      lMaxAscender;
LONG      lMaxDescender;
LONG      lLowerCaseAscent;
LONG      lLowerCaseDescent;
LONG      lInternalLeading;
LONG      lExternalLeading;
LONG      lAveCharWidth;
LONG      lMaxCharInc;
LONG      lEmInc;
LONG      lMaxBaselineExt;
SHORT     sCharSlope;
SHORT     sInlineDir;
SHORT     sCharRot;
USHORT    usWeightClass;
USHORT    usWidthClass;
SHORT     sXDeviceRes;
SHORT     sYDeviceRes;
SHORT     sFirstChar;
SHORT     sLastChar;
SHORT     sDefaultChar;
SHORT     sBreakChar;
SHORT     sNominalPointSize;
SHORT     sMinimumPointSize;
SHORT     sMaximumPointSize;
USHORT    fsType;
USHORT    fsDefn;
USHORT    fsSelection;
USHORT    fsCapabilities;
LONG      lSubscriptXSize;
LONG      lSubscriptYSize;
LONG      lSubscriptXOffset;
LONG      lSubscriptYOffset;
LONG      lSuperscriptXSize;
LONG      lSuperscriptYSize;
LONG      lSuperscriptXOffset;
LONG      lSuperscriptYOffset;
LONG      lUnderscoreSize;
LONG      lUnderscorePosition;
LONG      lStrikeoutSize;
LONG      lStrikeoutPosition;
SHORT     sKerningPairs;
SHORT     sFamilyClass;
LONG      lMatch;
LONG      FamilyNameAtom;
LONG      FaceNameAtom;
PANOSE    panose;
} FONTMETRICS;

typedef FONTMETRICS *PFONTMETRICS;

```



## Fields

### **szFamilyName**[FACESIZE] (CHAR)

Family name.

The family name of the font that describes the basic appearance of the font, for example, Times New Roman\*\* This string is null terminated, and therefore is limited to 31 characters in length. Longer names may be retrieved by using the *FamilyNameAtom* field to retrieve the full name from the System Atom table.

### **szFaceName**[FACESIZE] (CHAR)

Face name.

The typeface name that defines the particular font, for example, Times New Roman Bold Italic. This string is null terminated, and therefore is limited to 31 characters in length. Longer names may be retrieved by using the *FaceNameAtom* field to retrieve the full name from the System Atom table.

### **idRegistry** (USHORT)

Registry identifier.

The IBM registered number (or zero).

### **usCodePage** (USHORT)

Code page.

Defines the registered code page supported by the font. For example, the original IBM PC code page is 437. A value of 0 implies that the font may be used with any of the OS/2 supported code pages.

Where a font contains special symbols for which there is no registered code page, then code page 65400 is used.

### **lEmHeight** (LONG)

Em height.

The height of the Em square in world coordinate units. This corresponds to the point size for the font.

### **lXHeight** (LONG)

X height.

The nominal height above the baseline for lowercase characters (ignoring ascenders) in world coordinate units.

### **lMaxAscender** (LONG)

Maximum ascender.

The maximum height above the baseline reached by any part of any symbol in the font in world coordinate units. This field may exceed *lEmHeight*.

### **lMaxDescender** (LONG)

Maximum descender.

The maximum depth below the baseline reached by any part of any symbol in the font in world coordinate units. This field may exceed *lEmHeight*.

**ILowerCaseAscent** (LONG)

Lowercase ascent.

The maximum height above the baseline reached by any part of any lowercase (Latin unaccented “a” through “z”) symbol in the font in world coordinate units.

**ILowerCaseDescent** (LONG)

Lowercase descent.

The maximum depth below the baseline reached by any part of any lowercase (Latin unaccented “a” through “z”) symbol in the font in world coordinate units.

**InternalLeading** (LONG)

Internal leading.

The amount of space which, when subtracted from *IMaxAscender*, gives a font-design dependent, but glyph-set independent, measure of the distance above the baseline that characters extend. This calculation approximates the visual top to a row of characters without actually looking at the characters in the row.

For optimum results, this field should be used by applications to position the first line of a block of text by subtracting it from *IMaxAscender* and positioning the baseline that distance below whatever is above the text.

**Note:** This does not guarantee that characters will not overwrite information above them, but does give a font designer’s view of where to place the text. Collision should be tested for, and additional space allocated if necessary.

**IExternalLeading** (LONG)

External leading.

The amount of guaranteed white space advised by the font designer to appear between adjacent rows of text. This value may be zero.

**Note:** The fonts built in to Presentation Manager have zero in this field.

**IAveCharWidth** (LONG)

Average character width.

This is determined by multiplying the width of each lowercase character by a constant, adding the products, and then dividing by 1000. The letters involved in this, plus their constants, are as follows:

| Letter | Constant |
|--------|----------|
| a      | 64       |
| b      | 14       |
| c      | 27       |
| d      | 35       |
| e      | 100      |
| f      | 20       |
| g      | 14       |
| h      | 42       |
| i      | 63       |
| j      | 3        |
| k      | 6        |

|       |     |
|-------|-----|
| l     | 35  |
| m     | 20  |
| n     | 56  |
| o     | 56  |
| p     | 17  |
| q     | 4   |
| r     | 49  |
| s     | 56  |
| t     | 71  |
| u     | 31  |
| v     | 10  |
| w     | 18  |
| x     | 3   |
| y     | 18  |
| z     | 2   |
| space | 166 |

**Note:** For fixed pitch fonts, this value will be the same as the (A width + B width + C width) escapement of each character.

#### **IMaxCharInc (LONG)**

Maximum character increment.

The maximum character increment for the font in world coordinate units.

#### **IEmInc (LONG)**

Em increment.

The width of the Em square in world coordinate units. This corresponds to the point size of the font. When the horizontal device resolution equals the vertical device resolution this is equal to the em height.

#### **IMaxBaselineExt (LONG)**

Maximum baseline extent.

The maximum vertical space occupied by the font, in world coordinate units. This is the sum of *IMaxAscender* and *IMaxDescender* if both are positive. It is also the sum of *IInternalLeading* and *IEmHeight*.

One possible type of line spacing can be computed by adding *IMaxBaselineExt* to *IExternalLeading*. Such a line spacing, however, would be dependent on the glyph set included in the font. If a new version of the font should be made available, with new glyphs, then it is possible that this value will change because one of the new glyphs has gone above the previous *IMaxAscender* or below the previous *IMaxDescender*. More sophisticated applications will base line spacing on the point size (*IEmHeight*) of the font, which is an invariant of the font, multiplied by some factor (for example, 120%) plus any external leading.

This field may exceed *IEmHeight*.

#### **sCharSlope (SHORT)**

Character slope.

Defines the nominal slope for the characters of a font. The slope is defined in degrees

increasing clockwise from the vertical. An *italic* font is an example of a font with a nonzero slope.

**Note:** The units for this metric are degrees and minutes, encoded as shown in the following example:

180 degrees 59 minutes would be represented as :

|                 |                 |
|-----------------|-----------------|
| < byte 1 >      | < byte 2 >      |
| < Minutes >     | < Degrees >     |
| 0 0 1 1 1 0 1 1 | 1 0 1 1 0 1 0 0 |
| 59 min          | 180 degrees     |

### **sInlineDir** (SHORT)

Inline direction.

The direction in which the characters in the font are designed for viewing. The direction is defined in degrees increasing clockwise from the horizontal (left-to-right). Characters are added to a line of text in the inline direction.

**Note:** The units for this metric are degrees and minutes, encoded as shown in *sCharSlope*.

### **sCharRot** (SHORT)

Character rotation.

The rotation of the character glyphs with respect to the baseline, the angle increasing counter clockwise. This is the angle assigned by the font designer.

**Note:** The units for this metric are degrees and minutes, encoded as shown in *sCharSlope*.

### **usWeightClass** (USHORT)

Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

| <u>Value</u> | <u>Description</u> |
|--------------|--------------------|
| 1            | Ultra-light        |
| 2            | Extra-light        |
| 3            | Light              |
| 4            | Semi-light         |
| 5            | Medium (normal)    |
| 6            | Semi-bold          |
| 7            | Bold               |
| 8            | Extra-bold         |
| 9            | Ultra-bold         |

### **usWidthClass** (USHORT)

Width class.

Indicates the relative aspect ratio of the characters of the font in relation to the normal aspect ratio for this type of font:

| Value | Description     | % of normal width |
|-------|-----------------|-------------------|
| 1     | Ultra-condensed | 50                |
| 2     | Extra-condensed | 62.5              |
| 3     | Condensed       | 75                |
| 4     | Semi-condensed  | 87.5              |
| 5     | Medium (normal) | 100               |
| 6     | Semi-expanded   | 112.5             |
| 7     | Expanded        | 125               |
| 8     | Extra-expanded  | 150               |
| 9     | Ultra-expanded  | 200               |

#### **sXDeviceRes** (SHORT)

X-device resolution.

For bit-map fonts this is the resolution in the X direction of the intended target device, measured in pels per inch.

For outline fonts this is the number of notional units in the X direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

#### **sYDeviceRes** (SHORT)

Y-device resolution.

For bit-map fonts this is the resolution in the Y direction of the intended target device, measured in pels per inch.

For outline fonts this is the number of notional units in the Y direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

#### **sFirstChar** (SHORT)

First character.

The code point of the first character in the font.

#### **sLastChar** (SHORT)

Last character.

The code point of the last character in the font, expressed as an offset from *sFirstChar*.

All code points between the first and last character specified must be supported by the font.

#### **sDefaultChar** (SHORT)

Default character.

The code point that is used if a code point outside the range supported by the font is used, expressed as an offset from *sFirstChar*.

#### **sBreakChar** (SHORT)

Break character.

The code point that represents the "space" or "break" character for this font, expressed as an offset from *sFirstChar*. For example, if the first character is the space in code page 850, *sFirstChar* = 32, and *sBreakChar* = 0.

**sNominalPointSize** (SHORT)

Nominal point size.

For a bit-map font, this field contains the height of the font.

For an outline font, this field contains the height intended by the font designer. For example, some fonts are designed for text use in which case a value of 120 (12 point) would probably be placed in this field, whereas other fonts are designed for “display” use (“display” is typographer’s terminology for larger sizes). This is not the only size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

**sMinimumPointSize** (SHORT)

Minimum point size.

For a bit-map font, this field does not apply. For an outline font, this field contains the minimum height intended by the font designer. Note that this is not a restriction of the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

**sMaximumPointSize** (SHORT)

Maximum point size.

For a bit-map font, this field does not apply.

For an outline font, this field contains the maximum height intended by the font designer. Note that this is not a restriction of the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

**fsType** (USHORT)

Type indicators.

Contains this information:

|                   |   |
|-------------------|---|
| FM_TYPE_FIXED     | Characters in the font have the same fixed width.   |
| FM_TYPE_LICENSED  | Licensed (protected) font.  |
| FM_TYPE_KERNING   | Font contains kerning information.  |
| FM_TYPE_64K       | Font is larger than 64KB (KB equals 1024 bytes) in size. If the following two bits are false, the font is for single-byte code pages. One of the bits may be set. |
| FM_TYPE_DBCS      | Font is for double-byte code pages.   |
| FM_TYPE_MBCS      | Font is for mixed single- or double-byte code pages.  |
| FM_TYPE_FACETRUNC | Font <i>szFaceName</i> has been truncated.  |
| FM_TYPE_FAMTRUNC  | Font <i>szFamilyName</i> has been truncated.  |
| FM_TYPE_ATOMS     | The System Atom table atom values in <i>FamilyNameAtom</i> and in <i>FaceNameAtom</i> are valid.  |

**fsDefn** (USHORT)

Definition indicators.

Contains the following font definition data:

FM\_DEFN\_OUTLINE Font is a vector (outline) font; otherwise, it is a bit-map font.

FM\_DEFN\_GENERIC Font is in a format that can be used by the GPI; otherwise, it is a device font.

**fsSelection (USHORT)**

Selection indicators.

Contains information about the font patterns in the physical font.

**Note:** The flags do not reflect simulations applied to the physical font.

Possible values are:

|                       |  |
|-----------------------|--|
| FM_SEL_ITALIC         | True indicates that this font is designed as an italic font.   |
| FM_SEL_UNDERSCORE     | TRUE indicates that this font is designed with underscores included in each character.   |
| FM_SEL_NEGATIVE       | TRUE indicates that this font is designed with the background and foreground reversed.   |
| FM_SEL_OUTLINE        | TRUE indicates that this font is designed with outline (hollow) characters.  |
| FM_SEL_STRIKEOUT      | TRUE indicates that this font is designed with an overstrike through each character.   |
| FM_SEL_BOLD           | TRUE indicates that this font is designed with bold characters.  |
| FM_SEL_ISO9241_TESTED | This flag indicates that the font has been tested for compliance to ISO 9241. The presence of this flag doesn't indicate whether the font passed or failed, only that it was tested. |

**Note:** While the fonts were primarily tested for meeting the ISO standard, they have also been designed to meet the German standard DIN 66 234. Where the two standards differ, the fonts have been designed to meet the more stringent requirement.

**fsCapabilities (USHORT)**

Capabilities.

This attribute applies only to device fonts.

|              |   |
|--------------|---|
| FM_CAP_NOMIX | Characters may not be mixed with graphics.  |
| QUALITY      | The most significant byte may contain the following numeric value:<br><b>0</b> Undefined<br><b>1</b> DP quality<br><b>2</b> DP draft<br><b>3</b> Near Letter Quality<br><b>4</b> Letter Quality |

**ISubscriptXSize (LONG)**

Subscript x-size.

The horizontal size recommended by the font designer for subscripts for this font in world coordinate units.

**ISubscriptYSize (LONG)**

Subscript y-size.

The vertical size recommended by the font designer for subscripts for this font in world coordinate units.

**ISubscriptXOffset (LONG)**

Subscript x-offset.

The baseline x-offset recommended by the font designer for subscripts for this font in world coordinate units.

**ISubscriptYOffset (LONG)**

Subscript y-offset.

The baseline y-offset recommended by the font designer for subscripts for this font in world coordinate units.

**Note:** Positive numbers indicate an offset below the baseline.

**ISuperscriptXSize (LONG)**

Superscript x-size.

The horizontal size recommended by the font designer for superscripts for this font in world coordinate units.

**ISuperscriptYSize (LONG)**

Superscript y-size.

The vertical point size recommended by the font designer for superscripts for this font in world coordinate units.

**ISuperscriptXOffset (LONG)**

Superscript x-offset.

The baseline x-offset recommended by the font designer for superscripts for this font in world coordinate units.

**ISuperscriptYOffset (LONG)**

Superscript y-offset.

The baseline y-offset recommended by the font designer for superscripts for this font in world coordinate units.

**IUnderscoreSize (LONG)**

Underscore size.

The width (thickness) of the underscore stroke in world coordinate units. This describes the actual underscore in the font if FM\_SEL\_UNDERSCORE is also set. Otherwise it describes what the engine will simulate if underscore is requested in GpiCreateLogFont.

**IUnderscorePosition (LONG)**

Underscore position.

The position of the underscore stroke from the baseline in world coordinate units. This describes the actual underscore in the font if FM\_SEL\_UNDERSCORE is also set.



Otherwise it describes what the engine will simulate if underscore is requested in GpiCreateLogFont.

**Note:** Positive values indicate an offset below the baseline.

**IStrikeoutSize (LONG)**

Strikeout size.

The width of the strikeout stroke in world coordinate units. This describes the actual underscore in the font if FM\_SEL\_STRIKEOUT is also set. Otherwise it describes what the engine will simulate if overstrike is requested in GpiCreateLogFont.

**IStrikeoutPosition (LONG)**

Strikeout position.

The position of the strikeout stroke relative to the baseline in world coordinate units. This describes the actual underscore in the font if FM\_SEL\_STRIKEOUT is also set. Otherwise it describes what the engine will simulate if overstrike is requested in GpiCreateLogFont.

**sKerningPairs (SHORT)**

Kerning pairs.

The number of kerning pairs in the kerning pair table.

**sFamilyClass (SHORT)**

Font family design classification.

This value contains a font class and its subclass.

**IMatch (LONG)**

Matched font identity.

This uniquely identifies the font for a given device and device driver combination. A positive match number signifies that the font is a generic (engine) font while a negative number indicates a device font (a native or downloadable font). This value should not be used to identify a font across system boundaries.

**FamilyNameAtom (LONG)**

Font family name atom.

This value contains the atom identifier for the font family name in the System Atom Table.

**FaceNameAtom (LONG)**

Font facename atom.

This value contains the atom identifier for the font face name in the System Atom Table.

**panose (PANOSE)**

Panose font descriptor.

This is the Panose descriptor identifying the visual characteristics of the font.

---

## FRAMECDATA

Frame-control data structure.

### Syntax

```
typedef struct _FRAMECDATA {
    USHORT      cb;
    ULONG       flCreateFlags;
    USHORT      hmodResources;
    USHORT      idResources;
} FRAMECDATA;

typedef FRAMECDATA *PFRAMECDATA;
```

### Fields

**cb** (USHORT)  
Length.

**flCreateFlags** (ULONG)  
Frame-creation flags.

Possible values are described in the following list:

- FCF\_TITLEBAR
- FCF\_SYSMENU
- FCF\_MENU
- FCF\_SIZEBORDER
- FCF\_MINBUTTON
- FCF\_MAXBUTTON
- FCF\_MINMAX
- FCF\_VERTSCROLL
- FCF\_HORZSCROLL
- FCF\_DLGBORDER
- FCF\_BORDER
- FCF\_SHELLPOSITION
- FCF\_TASKLIST
- FCF\_NOBYTEALIGN
- FCF\_NOMOVEWITHOWNER
- FCF\_ICON
- FCF\_ACCELTABLE
- FCF\_SYSMODAL
- FCF\_SCREENALIGN
- FCF\_MOUSEALIGN
- FCF\_HIDEBUTTON
- FCF\_HIDEMAX
- FCF\_AUTOICON
- FCF\_DBE\_APPSTAT

FCF\_STANDARD

The standard setting is equivalent to setting FCF\_TITLEBAR, FCF\_SYSMENU, FCF\_MENU, FCF\_SIZEBORDER, FCF\_MINMAX, FCF\_ICON, FCF\_ACCELTABLE, FCF\_SHELLPOSITION, and FCF\_TASKLIST.

**hmodResources** (USHORT)

Identifier of required resource.

This is supplied in an environment-dependent manner.

**idResources** (USHORT)

Resource identifier.

---

## GRADIENTL

Direction-vector structure.

### Syntax

```
typedef struct _GRADIENTL {  
    LONG    x;  
    LONG    y;  
} GRADIENTL;  
  
typedef GRADIENTL *PGRADIENTL;
```

### Fields

**x** (LONG)

X-component of direction.

**y** (LONG)

Y-component of direction.

---

## HAB

Anchor-block handle.

### Syntax

```
typedef LHANDLE HAB;
```

---

## HACCEL

Accelerator-table handle.

### Syntax

```
typedef LHANDLE HACCEL;
```

---

## HAPP

Handle of an application.

### Syntax

```
typedef LHANDLE HAPP;
```

---

## HATOMTBL

Atom-table handle.

### Syntax

```
typedef LHANDLE HATOMTBL;
```

---

## HBITMAP

Bit-map handle.

### Syntax

```
typedef LHANDLE HBITMAP;
```

---

## HDC

Device-context handle.

### Syntax

```
typedef LHANDLE HDC;
```

---

## HCINFO

Hardcopy-capabilities structure.

### Syntax

```
typedef struct _HCINFO {  
    CHAR    szFormname[32];  
    LONG    cx;  
    LONG    cy;  
    LONG    xLeftClip;  
    LONG    yBottomClip;  
    LONG    xRightClip;  
    LONG    yTopClip;  
    LONG    xPels;  
    LONG    yPels;  
    LONG    flAttributes;  
} HCINFO;  
  
typedef HCINFO *PHCINFO;
```

### Fields

**szFormname[32]** (CHAR)

Form name.

**cx** (LONG)

Width (left-to-right) in millimeters.

**cy** (LONG)

Height (top-to-bottom) in millimeters.

**xLeftClip** (LONG)

Left clip limit in millimeters.

**yBottomClip** (LONG)

Bottom clip limit in millimeters.

**xRightClip** (LONG)

Right clip limit in millimeters.

**yTopClip** (LONG)

Top clip limit in millimeters.

**xPels** (LONG)

Number of pels between left and right clip limits.

**yPels** (LONG)

Number of pels between bottom and top clip limits.

**flAttributes** (LONG)

Attributes of the form identifier.

|                  |   |
|------------------|---|
| HCAPS_SELECTABLE | The form is installed on the printer as given by the printer properties dialog. It is available from an alternate form source without operator intervention. If the form does not have this bit set, and is used (if the user selects it), a “forms mismatch” error is generated by the printer object. |
| HCAPS_CURRENT    | The form is the one currently selected by the DevOpenDC DEVOPENSTRUC <i>pdriv</i> field (the job properties).   |

---

**HDDF**

Dynamic data formatting handle.

**Syntax**

```
typedef VOID * HDDF;
```

## HELPINIT

Help Manager initialization structure.

### Syntax

```
typedef struct _HELPINIT {
    ULONG          cb;
    ULONG          ulReturnCode;
    PSZ            pszTutorialName;
    HELPTABLE      htHelpTable;
    HMODULE        hmodHelpTableModule;
    HMODULE        hmodAccelerActionbarModule;
    ULONG          idAccelerTable;
    ULONG          idActionbar;
    PSZ            pszHelpWindowTitle;
    ULONG          fShowPanelId;
    PSZ            pszHelpLibraryName;
} HELPINIT;

typedef HELPINIT *PHELPINIT;
```

### Fields

#### **cb** (ULONG)

Count of bytes of the initialization structure.

#### **ulReturnCode** (ULONG)

Value returned by the Help Manager from initialization.

0 Initialization was successful.

#### **pszTutorialName** (PSZ)

Indicates to the Help Manager that the application has a tutorial program.

NULL The application either does not have a tutorial program, or the tutorial name is specified in each help panel definition.

Other Default tutorial name.

#### **htHelpTable** (HELPTABLE)

Help table.

The help table or the identity of the help table. If this is the identity of the help table in a resource file, the low-order word contains the identity of the table and the high-order word must be 0xFFFF.

The help table associates each application window with its help subtable and the identity of its extended help panel.

**hmodHelpTableModule (HMODULE)**

Resource file identity.

If the *htHelpTable* contains the identity of the help table, this field identifies the module handle returned by the *DosLoadModule* call by which the application loaded the resource file.

NULL The resource file containing the help table was appended to the application's .EXE file.

Other Resource file identity.

**hmodAccelActionBarModule (HMODULE)**

Handle of the containing DLL.

The handle of the DLL which contains the accelerator table and action bar template to be used by the Help Manager.

NULL Use the default action bar and accelerator table defined by the Help Manager.

Other Handle of the DLL.

**idAccelTable (ULONG)**

Identity of the accelerator table.

The accelerator table resides in the DLL provided in the *hmodAccelActionBarModule* field.

NULL Use the default accelerator table.

Other Identity of the accelerator table.

**idActionBar (ULONG)**

Identity of the action bar template used by the Help Manager.

The action bar template resides in the DLL provided in the *hmodAccelActionBarModule* field.

NULL Use the default action bar.

Other Identity of the action bar.

**pszHelpWindowTitle (PSZ)**

Window title for the main help window of this help instance.

**fShowPanelId (ULONG)**

Show panel identity indicator.

The constants corresponding to the panel identity flags are in the *PMHELP.H* include file.

CMIC\_SHOW\_PANEL\_ID Show the panel identity on a help panel.

CMIC\_HIDE\_PANEL\_ID Do not show the panel identity on a help panel.



### **pszHelpLibraryName (PSZ)**

Help panel library names.

The names of the help panel libraries that the Help Manager searches on each help request. The names must be separated by a blank.

The Help Manager looks for the libraries in the path set by the HELP environment variable. If the library is not found, the Help Manager will look for the libraries in the current directory.

---

## **HELPSUBTABLE**

Help subtable.

A help subtable is an array of records, preceded by a value that specifies the size of each help-subtable record.

### **Syntax**

```
typedef USHORT _HELPSUBTABLE {
    USHORT      usSubitemSize;
    USHORT      HelpSubTableEntry[];
} HELPSUBTABLE;
```

The first entry in the help subtable indicates the size of the records that follow in the subtable. Each of the following entries in the help subtable is a record that consists of a Field ID parameter, a Help Panel ID parameter, and an optional array of application-related USHORT integers. The minimum number of words in the record is two: the Field ID and the Help Panel ID. The last record in the subtable must be a NULL entry.

The Field ID is the symbolic constant for a field from which the user can request help. The Field ID can identify a control, a menu item, or a message box, and must be unique across the help subtable. The value 0xFFFF is reserved for use by the Help Manager.

The Help Panel ID is the resource ID (res) of the contextual help panel to be associated with the field in the Field ID parameter. This is the panel to be displayed when the user requests help for the field.

The optional array of USHORT integers is ignored by the Help Manager and can be used to store information of relevance to the application.

There can be a maximum of 16,000 help subtables for a given help instance and each subtable can have a maximum of 64K bytes of data.

The following figure contains the declaration of a help subtable that contains only Field IDs and Help Panel IDs. In this subtable, each of the records after the size entry consists of 1 Field ID and 1 Help Panel ID for a size of 2. Note that the last record is filled with NULLs (0) to indicate the end of the array.

```
HELPSUBTABLE HelpSubTable[] =
{
    2,                /* Size of each record */
    FIELD_ID_1, IDRES_HELP1, /* The first record */
    FIELD_ID_2, IDRES_HELP2, /* The second record */
    FIELD_ID_3, IDRES_HELP3, /* The third record */
    FIELD_ID_4, IDRES_HELP4, /* The fourth record */
    FIELD_ID_5, IDRES_HELP5, /* The fifth record */
    FIELD_ID_6, IDRES_HELP6, /* The sixth record */
    0,                /* NULL record == end of the array */
}
```

## Fields

### **usSubitemSize** (USHORT)

Size of each record of the help subtable.

This entry defines the number of parameters in each record in the rest of the help subtable. The minimum number of words in each record is two.

2        The minimum number of words in each record of the help subtable. This value is used when each help subtable record consists only of a Field ID and Help Panel ID.

Other    This value is used when a help subtable record consists of a Field ID, Help Panel ID, and an array of application-related USHORT integers.

### **HelpSubTableEntry[]** (USHORT)

Help subtable records.

This is the array of help subtable records, each of which contains a Field ID, a Help Panel ID, and an optional array of application-related USHORT integers. The last record of the array must be a NULL entry.

---

## HELPTABLE

Help table.

This is a collection of help table entries, each of which has the structure defined below, the last entry of the collection being a NULL structure.

### Syntax

```
typedef struct _HELPTABLE {
    USHORT          idAppWindow;
    PHELPSUBTABLE  phstHelpSubTable;
    USHORT          idExtPanel;
} HELPTABLE;

typedef HELPTABLE *PHELPTABLE;
```

### Fields

#### **idAppWindow** (USHORT)

Application window identity.

#### **phstHelpSubTable** (PHELPSUBTABLE)

Help subtable for this application window.

#### **idExtPanel** (USHORT)

Identity of the extended help panel for the application window.

---

## HENUM

Window-enumeration handle.

### Syntax

```
typedef LHANDLE HENUM;
```

---

## HEV

32-bit value used as an event semaphore handle.

### Syntax

```
typedef ULONG HEV;
```

---

## **HINI**

Initialization-file handle.

### **Syntax**

```
typedef LHANDLE HINI;
```

---

## **HLIB**

Library handle.

### **Syntax**

```
typedef HMODULE HLIB;
```

---

## **HMF**

Metafile handle.

### **Syntax**

```
typedef LHANDLE HMF;
```

---

## **HMODULE**

Module handle.

### **Syntax**

```
typedef LHANDLE HMODULE;
```

---

**HMQ**

Message-queue handle.

**Syntax**

```
typedef LHANDLE HMQ;
```

---

**HMTX**

32-bit value used as a mutex semaphore handle.

**Syntax**

```
typedef ULONG HMTX;
```

---

**HMUX**

32-bit value used as a muxwait semaphore handle.

**Syntax**

```
typedef ULONG HMUX;
```

---

**HOBJECT**

Workplace object handle.

**Syntax**

```
typedef LHANDLE HOBJECT;
```

---

## **HPOINTER**

Pointer handle.

### **Syntax**

```
typedef LHANDLE HPOINTER;
```

---

## **HPROGRAM**

Program handle.

### **Syntax**

```
typedef LHANDLE HPROGRAM;
```

---

## **HPS**

Presentation-space handle.

### **Syntax**

```
typedef LHANDLE HPS;
```

---

## **HRGN**

Region handle.

### **Syntax**

```
typedef LHANDLE HRGN;
```

---

## **HSAVEWP**

Frame window-repositioning process handle.

### **Syntax**

```
typedef LHANDLE HSAVEWP;
```

---

## **HSEM**

Semaphore handle.

### **Syntax**

```
typedef VOID * HSEM;
```

---

## **HSPL**

Spooler handle.

### **Syntax**

```
typedef LHANDLE HSPL;
```

---

## **HSTR**

String handle.

### **Syntax**

```
typedef LHANDLE HSTR;
```

---

## HSWITCH

Switch-list entry handle.

### Syntax

```
typedef LHANDLE HSWITCH;
```

---

## HWND

Window handle.

### Syntax

```
typedef LHANDLE HWND;
```

---

## ICONINFO

Icon information data structure.

### Syntax

```
typedef struct _ICONINFO {
    ULONG      cb;
    ULONG      fFormat;
    PSZ        pszFileName;
    HMODULE    hmod;
    ULONG      resid;
    ULONG      cbIconData;
    PVOID      pIconData;
} ICONINFO;

typedef ICONINFO *PICONINFO;
```

### Fields

**cb** (ULONG)

Length of ICONINFO structure.



**fFormat (ULONG)**

Indicates from where the icon resides.

Possible values are:

|               |                          |
|---------------|--------------------------|
| ICON_FILE     | Icon file supplied.      |
| ICON_RESOURCE | Icon resource supplied.  |
| ICON_DATA     | Icon data supplied.      |
| ICON_CLEAR    | Go back to default icon. |

**pszFileName (PSZ)**

Name of file containing icon data.

This value is ignored if fFormat is not equal to to ICON\_FILE.

**hmod (HMODULE)**

Module containing the icon resource.

This value is ignored if fFormat is not equal to to ICON\_RESOURCE.

**resid (ULONG)**

Identity of icon resource.

This value is ignored if fFormat is not equal to to ICON\_RESOURCE.

**cbIconData (ULONG)**

Length of icon data in bytes.

This value is ignored if fFormat is not equal to to ICON\_DATA.

**pIconData (PVOID)**

Pointer to buffer containing icon data.

This value is ignored if fFormat is not equal to to ICON\_DATA.

---

**IMAGEBUNDLE**

Image-attributes bundle structure.

**Syntax**

```
typedef struct _IMAGEBUNDLE {
    LONG        lColor;
    LONG        lBackColor;
    USHORT      usMixMode;
    USHORT      usBackMixMode;
} IMAGEBUNDLE;

typedef IMAGEBUNDLE *PIMAGEBUNDLE;
```

## Fields

### **IColor** (LONG)

Image foreground color.

### **IBackColor** (LONG)

Image background color.

### **usMixMode** (USHORT)

Image foreground-mix mode.

### **usBackMixMode** (USHORT)

Image background-mix mode.

---

## **IPT**

Insertion point for multi-line entry field.

## **Syntax**

```
typedef LONG IPT;
```

---

## **KERNINGPAIRS**

Kerning-pair records structure.

## **Syntax**

```
typedef struct _KERNINGPAIRS {  
    SHORT    sFirstChar;  
    SHORT    sSecondChar;  
    LONG     lKerningAmount;  
} KERNINGPAIRS;  
  
typedef KERNINGPAIRS *PKERNINGPAIRS;
```

## Fields

### **sFirstChar** (SHORT)

First character of pair.

### **sSecondChar** (SHORT)

Second character of pair.

### **lKerningAmount** (LONG)

Amount of kerning for this pair.

---

## LBOXINFO

List box information structure.

### Syntax

```
typedef struct _LBOXINFO {
    LONG        lItemIndex;
    ULONG       ulItemCount;
    ULONG       reserved;
    ULONG       reserved2;
} LBOXINFO;

typedef LBOXINFO *PLBOXINFO;
```

### Fields

#### lItemIndex (LONG)

Index of the item to insert after.

Possible values are described in the following list:

|                    |  |
|--------------------|--|
| LIT_ENT            | Add items to the end of the list.  |
| LIT_SORTASCENDING  | Add items to the list and sort the complete list in ascending order.                       |
| LIT_SORTDESCENDING | Add items to the list and sort the complete list in descending order.                      |
| Other              | Add the items to the list after the specified zero-based index. Valid range is 0 to 32767. |

#### ulItemCount (ULONG)

Number of items to be inserted into the list.

A maximum of 32768 can be inserted into the list at one time.

#### reserved (ULONG)

Reserved value, must be 0.

#### reserved2 (ULONG)

Reserved value, must be 0.

---

## LHANDLE

The handle of a resource.

### Syntax

```
typedef unsigned long LHANDLE;
```

---

## LINEBUNDLE

Line-attributes bundle structure.

### Syntax

```
typedef struct _LINEBUNDLE {  
    LONG        lColor;  
    LONG        lBackColor;  
    USHORT      usMixMode;  
    USHORT      usBackMixMode;  
    FIXED       fxWidth;  
    LONG        lGeomWidth;  
    USHORT      usType;  
    USHORT      usEnd;  
    USHORT      usJoin;  
    USHORT      usReserved;  
} LINEBUNDLE;  
  
typedef LINEBUNDLE *PLINEBUNDLE;
```

### Fields

#### **IColor** (LONG)

Line foreground color.

#### **IBackColor** (LONG)

Line background color.

#### **usMixMode** (USHORT)

Line foreground-mix mode.

#### **usBackMixMode** (USHORT)

Line background-mix mode.

#### **fxWidth** (FIXED)

Line width.

#### **IGeomWidth** (LONG)

Geometric line width.

**usType** (USHORT)

Line type.

**usEnd** (USHORT)

Line end.

**usJoin** (USHORT)

Line join.

**usReserved** (USHORT)

Reserved.

---

## LONG

Signed integer in the range -2 147 483 648 through 2 147 483 647.

### Syntax

```
#define LONG long
```

**Note:** Where this data type represents a graphic coordinate in world or model space, its value is restricted to -134 217 728 through 134 217 727.

A graphic coordinate in device or screen coordinates is restricted to -32 768 through 32 767.

The value of a graphic coordinate may be further restricted by any transforms currently in force, including the positioning of the origin of the window on the screen. In particular, coordinates in world or model space must not generate coordinate values after transformation (that is, in device or screen space) outside the range -32 768 through 32 767.

---

## MARKERBUNDLE

Marker-attributes bundle structure.

### Syntax

```
typedef struct _MARKERBUNDLE {  
    LONG        IColor;  
    LONG        IBackColor;  
    USHORT      usMixMode;  
    USHORT      usBackMixMode;  
    USHORT      usSet;  
    USHORT      usSymbol;  
    SIZEF       sizfxCell;  
} MARKERBUNDLE;  
  
typedef MARKERBUNDLE *PMARKERBUNDLE;
```

### Fields

#### **IColor** (LONG)

Marker foreground color.

#### **IBackColor** (LONG)

Marker background color.

#### **usMixMode** (USHORT)

Marker foreground-mix mode.

#### **usBackMixMode** (USHORT)

Marker background-mix mode.

#### **usSet** (USHORT)

Marker set.

#### **usSymbol** (USHORT)

Marker symbol.

#### **sizfxCell** (SIZEF)

Marker cell.

---

## MATRIXLF

Matrix-elements structure.

### Syntax

```
typedef struct _MATRIXLF {  
    FIXED    fxM11;  
    FIXED    fxM12;  
    LONG     IM13;  
    FIXED    fxM21;  
    FIXED    fxM22;  
    LONG     IM23;  
    LONG     IM31;  
    LONG     IM32;  
    LONG     IM33;  
} MATRIXLF;  
  
typedef MATRIXLF *PMATRIXLF;
```

### Fields

#### **fxM11** (FIXED)

First element of first row.

#### **fxM12** (FIXED)

Second element of first row.

#### **IM13** (LONG)

Third element of first row.

#### **fxM21** (FIXED)

First element of second row.

#### **fxM22** (FIXED)

Second element of second row.

#### **IM23** (LONG)

Third element of second row.

#### **IM31** (LONG)

First element of third row.

#### **IM32** (LONG)

Second element of third row.

#### **IM33** (LONG)

Third element of third row.

---

## MB2D

Array of button definitions.

### Syntax

```
typedef struct _MB2D {
    CHAR        achText[MAX_MB2DTEXT+1];
    ULONG       idButtons;
    ULONG       flStyle;
} MB2D;

typedef MB2D *PMB2D;
```

### Fields

**achText[MAX\_MB2DTEXT+1]** (CHAR)

Text of the button.

For example, "Cancel."

Currently, MAX\_MB2DTEXT is equal to 70.

**idButtons** (ULONG)

Button Id returned when selected.

**flStyle** (ULONG)

Button style flags.

These style flags may be ORed with internal styles.

---

## MB2INFO

Button information block.

### Syntax

```
typedef struct _MB2INFO {
    ULONG       cb;
    HPOINTER    hIcon;
    ULONG       cButtons;
    ULONG       flStyle;
    HWND        hwndNotify;
    MB2D        mb2d[1];
} MB2INFO;

typedef MB2INFO *PMB2INFO;
```



## Fields

### **cb** (ULONG)

Current size of the structure.

### **hIcon** (HPOINTER)

Icon handle.

### **cButtons** (ULONG)

Number of buttons.

### **flStyle** (ULONG)

Icon style flags.

Possible values are described in the following list:

**MB\_APPLMODAL** Message box is application modal. This is the default case. Its owner is disabled; therefore, do not specify the owner as the parent if this option is used.

**MB\_ERROR** Message box contains a stop sign with a white background.

**MB\_ICONASTERISK** Message box contains a asterisk icon.

**MB\_CUSTOMICON** Message box contains a custom icon specified in *hIcon*.

**MB\_ICONEXCLAMATION** Message box contains a exclamation point icon.

**MB\_ICONHAND** Message box contains a hand icon.

**MB\_ICONQUERY** Message box contains a question mark in a box.

**MB\_ICONQUESTION** Message box contains a question mark icon.

**MB\_INFORMATION** Message box contains a black "i" in a box.

**MB\_MOVEABLE** Message box is moveable.

The message box is displayed with a title bar and a system menu, showing only the Move, Close, and Task Manager choices, which can be selected either by use of the pointing device or by accelerator keys.

**MB\_NOICON** Message box does not contain an icon.

**MB\_NONMODAL** Message box is nonmodal (the program continues after displaying the nonmodal message box).

The message box remains visible until the owner window destroys it. Two notification messages, **WM\_MSGBOXINIT** and **WM\_MSGBOXDISMISS**, are used to support this non-modality.

**MB\_SYSTEMMODAL** Message box is system modal.

**MB\_WARNING** Message box contains a black "!" in a box.

**hwndNotify** (HWND)  
Owner notification handle.

**mb2d[1]** (MB2D)  
Array of button definitions.

---

## MENUITEM

Menu item.

### Syntax

```
typedef struct _MENUITEM {  
    SHORT      iPosition;  
    USHORT     afStyle;  
    USHORT     afAttribute;  
    USHORT     id;  
    HWND       hwndSubMenu;  
    ULONG      hItem;  
} MENUITEM;  
  
typedef MENUITEM *PMENUITEM;
```

### Fields

**iPosition** (SHORT)  
Position.

**afStyle** (USHORT)  
Style.

**afAttribute** (USHORT)  
Attribute.

**id** (USHORT)  
Identity.

**hwndSubMenu** (HWND)  
Submenu.

**hItem** (ULONG)  
Item.

## MINIRECORDCORE

Structure that contains information for smaller records than those defined by the RECORDCORE data structure. This data structure is used if the CCS\_MINIRECORDCORE style bit is specified when a container is created.

### Syntax

```
typedef struct _MINIRECORDCORE {
    ULONG          cb;
    ULONG          flRecordAttr;
    POINTL        ptlIcon;
    struct _MINIRECORDCORE *preccNextRecord;
    PSZ           pszIcon;
    HPOINTER      hptrIcon;
} MINIRECORDCORE;

typedef MINIRECORDCORE *PMINIRECORDCORE;
```

### Fields

#### cb (ULONG)

Structure size.

The size (in bytes) of the MINIRECORDCORE structure.

#### flRecordAttr (ULONG)

Attributes of container records.

Contains any or all of the following:

|                    |  |
|--------------------|--|
| CRA_COLLAPSED      | Specifies that a record is collapsed.                                |
| CRA_CURSORED       | Specifies that a record will be drawn with a selection cursor.       |
| CRA_DROPONABLE     | Specifies that a record can be a target for direct manipulation.     |
| CRA_EXPANDED       | Specifies that a record is expanded.                                 |
| CRA_FILTERED       | Specifies that a record is filtered, and therefore hidden from view. |
| CRA_INUSE          | Specifies that a record will be drawn with in-use emphasis.          |
| CRA_RECORDREADONLY | Prevents a record from being edited directly.                        |
| CRA_SELECTED       | Specifies that a record will be drawn with selected-state emphasis.  |
| CRA_TARGET         | Specifies that a record will be drawn with target emphasis.          |

**ptlIcon (POINTL)**

Record position.

Position of a container record in the icon view.

**preccNextRecord (struct \_MINIRECORDCORE \*)**

Pointer to the next linked record.

**pszIcon (PSZ)**

Record text.

Text for the container record.

**hptrIcon (HPOINTER)**

Record icon.

Icon that is displayed for the container record.

---

**MLE\_SEARCHDATA**

Search structure for multiline entry field.

**Syntax**

```
typedef struct _SEARCH {
    USHORT      cb;
    PCHAR       pchFind;
    PCHAR       pchReplace;
    SHORT       cchFind;
    SHORT       cchReplace;
    IPT         iptStart;
    IPT         iptStop;
    USHORT      cchFound;
} MLE_SEARCHDATA;

typedef MLE_SEARCHDATA *PMLE_SEARCHDATA;
```

**Fields****cb (USHORT)**

Size of structure.

**pchFind (PCHAR)**

String to search for.

**pchReplace (PCHAR)**

String to replace with.

**cchFind (SHORT)**

Length of *pchFind* string.

**cchReplace** (SHORT)

Length of *pchReplace* string.

**iptStart** (IPT)

Point at which to start search, or point where string was found.

Non-negative Point at which to start search.

Negative Start search from current cursor location.

**iptStop** (IPT)

Point at which to stop search.

Non-negative Point at which to stop search.

Negative Stop search at end of text.

**cchFound** (USHORT)

Length of string found at *iptStart*.

---

**MLEMARGSTRUCT**

Multiline entry-field margin information.

**Syntax**

```
typedef struct _MLEMARGSTRUCT {
USHORT      afMargins;
USHORT      usMouMsg;
IPT         iptNear;
} MLEMARGSTRUCT;

typedef MLEMARGSTRUCT *PMARGSTRUCT;
```

## Fields

### **afMargins** (USHORT)

Margin in which the event occurred.

The left and right margins are defined as including the corners at the top and bottom, and the top and bottom margins are contained between them. Therefore, the corners are included in the sides.

MLFMARGIN\_LEFT  
MLFMARGIN\_RIGHT  
MLFMARGIN\_TOP  
MLFMARGIN\_BOTTOM

### **usMouMsg** (USHORT)

Message identity of the original mouse event.

### **iptNear** (IPT)

Insertion point nearest to the margin event.

---

## MLECTLDATA

Multiline entry-field (MLE) control data structure.

## Syntax

```
typedef struct MLECTLDATA {
USHORT      cbCtlData;
USHORT      afIEFormat;
ULONG       cchText;
IPT         iptAnchor;
IPT         iptCursor;
LONG        cxFormat;
LONG        cyFormat;
ULONG       afFormatFlags;
} MLECTLDATA;

typedef MLECTLDATA *PMLECTLDATA;
```

## Fields

### **cbCtlData** (USHORT)

Length of control data in bytes.

### **afIEFormat** (USHORT)

Import/export format.

This sets the initial import/export format. Setting this value via control data is considered identical to setting it through the MLM\_FORMAT message. The same constants apply here. The default is MLE\_CFTEXT.

**cchText (ULONG)**

Text limit.

The maximum amount of text allowed in the MLE. This value is interpreted identically to the parameter of MLM\_SETTEXTLIMIT. A negative value indicates that the length is considered unbounded.

**iptAnchor (IPT)**

Selection anchor point.

**iptCursor (IPT)**

Selection cursor point.

The *iptAnchor* and *iptCursor* parameters identify the beginning and ending points, respectively, of the selection. These values may range from 0 through the length of the text. The default is 0,0 and can be indicated by entering 0,0.

**cxFormat (LONG)**

Formatting-rectangle width in pels.

**cyFormat (LONG)**

Formatting-rectangle height in pels.

The *cxFormat* and *cyFormat* parameters identify the dimensions in pels of the formatting rectangle, as can be set by the MLM\_SETFORMATRECT message. These values are considered identical to the two fields in the format rectangle structure referenced in that message, and the interpretation of the values in these fields is governed by the *afFormatFlags* field.

The default is the window size in both dimensions, and can be indicated by 0 values.

**afFormatFlags (ULONG)**

Format flags.

These flags govern the interpretation of the *cxFormat* and *cyFormat* fields, just as in the MLM\_SETFORMATRECT message. The flag values defined there are also valid in this field. The default is unlimited in both directions, and is of varying size to match the window size.

---

## MPARAM

4-byte message-dependent parameter structure.

### Syntax

```
typedef VOID * MPARAM;
```

Certain elements of information, placed into the parameters of a message, have data types that do not use all 4 bytes of this data type. The rules governing these cases are:

- BOOL**     The value is contained in the low word and the high word is 0.
- SHORT**    The value is contained in the low word and its sign is extended into the high word.
- USHORT**   The value is contained in the low word and the high word is 0.
- NULL**     The entire 4 bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

---

## MQINFO

Message-queue information structure.

### Syntax

```
typedef struct _MQINFO {
    ULONG        cb;
    PID          pid;
    TID          tid;
    ULONG        cmsgs;
    PVOID        pReserved;
} MQINFO;

typedef MQINFO *PMQINFO;
```

### Fields

- cb** (ULONG)  
Length of structure.
- pid** (PID)  
Process identity.
- tid** (TID)  
Thread identity.



**cmsgs** (ULONG)

Message count.

**pReserved** (PVOID)

Reserved.

---

## MRESULT

4-byte message-dependent reply parameter structure.

Certain elements of information, placed into the parameters of a message, have data types that do not use all 4 bytes of this data type. The rules governing these cases are:

**BOOL** The value is contained in the low word and the high word is 0.

**SHORT** The value is contained in the low word and its sign is extended into the high word.

**USHORT** The value is contained in the low word and the high word is 0.

**NULL** The entire 4 bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

### Syntax

```
typedef VOID * MRESULT;
```

---

## NOTIFYDELTA

Structure that contains information about the placement of delta information for a container. This structure is used in the CN\_QUERYDELTA container notification code only. See "CN\_QUERYDELTA" on page 22-27 for information about that notification code.

### Syntax

```
typedef struct _NOTIFYDELTA {  
    HWND      hwndCnr;  
    ULONG     fDelta;  
} NOTIFYDELTA;  
  
typedef NOTIFYDELTA *PNOTIFYDELTA;
```

## Fields

### hwndCnr (HWND)

Container control handle.

### fDelta (ULONG)

Placement of delta information. The values can be:

|               |   |
|---------------|---|
| CMA_DELTATOP  | The record that represents the delta value scrolls into view at the top of the client area.   |
| CMA_DELTABOT  | The record that represents the delta value scrolls into view at the bottom of the client area.  |
| CMA_DELTAHOME | The container scrolls to the beginning of the list of all container records that are available to be inserted into the container, such as the first record in a database. |
| CMA_DELTAEND  | The container scrolls to the end of the list of all container records that are available to be inserted into the container, such as the last record in a database.        |

---

## NOTIFYRECORDEMPHASIS

Structure that contains information about emphasis that is being applied to a container record. This structure is used in the CN\_EMPHASIS container notification code only. See "CN\_EMPHASIS" on page 22-20 for information about that notification code.

## Syntax

```
typedef struct _NOTIFYRECORDEMPHASIS {
    HWND          hwndCnr;
    RECORDCORE    pRecord;
    ULONG         fEmphasisMask;
} NOTIFYRECORDEMPHASIS;

typedef NOTIFYRECORDEMPHASIS *PNOTIFYRECORDEMPHASIS;
```

## Fields

### hwndCnr (HWND)

Container control handle.

### pRecord (RECORDCORE)

Pointer to a RECORDCORE data structure whose emphasis attribute has been changed.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of RECORDCORE in all applicable data structures and messages.

### **fEmphasisMask (ULONG)**

Changed emphasis attributes.

Specifies the emphasis attribute or attributes that changed in the container record. The following states can be combined with a logical OR operator (|):

- CRA\_CURSORED
- CRA\_INUSE
- CRA\_SELECTED.

---

## **NOTIFYRECORDENTER**

Structure that contains information about the input device that is being used with the container control. This structure is used in the CN\_ENTER container notification code only. See "CN\_ENTER" on page 22-22 for information about that notification code.

### **Syntax**

```
typedef struct _NOTIFYRECORDENTER {
    HWND          hwndCnr;
    ULONG         fKey;
    PRECORDCORE  pRecord;
} NOTIFYRECORDENTER;

typedef NOTIFYRECORDENTER *PNOTIFYRECORDENTER;
```

### **Fields**

#### **hwndCnr (HWND)**

Container control handle.

#### **fKey (ULONG)**

Flag.

Flag that determines whether the Enter key was pressed or the select button was double-clicked.

TRUE     The Enter key was pressed.  
FALSE    The select button was double-clicked.

## **pRecord** (RECORDCORE)

Pointer to the RECORDCORE data structure over which an action occurred.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

- If a user presses the Enter key, a pointer to the record with the selection cursor is returned.
- If a user double-clicks the select button when the pointer of the pointing device is over a record, a pointer to the record is returned.
- If a user double-clicks the select button when the pointer of the pointing device is over white space, NULL is returned.

---

## **NOTIFYSCROLL**

Structure that contains information about scrolling a container control window. This structure is used in the CN\_SCROLL container notification code only. See "CN\_SCROLL" on page 22-29 for information about that notification code.

### **Syntax**

```
typedef struct _NOTIFYSCROLL {
    HWND      hwndCnr;
    LONG      lScrollInc;
    ULONG     fScroll;
} NOTIFYSCROLL;

typedef NOTIFYSCROLL *PNOTIFYSCROLL;
```

### **Fields**

#### **hwndCnr** (HWND)

Container control handle.

#### **lScrollInc** (LONG)

Scroll amount.

Amount (in pixels) by which the window scrolled.

#### **fScroll** (ULONG)

Scroll flags.

Flags that show the direction in which the window scrolled and the window that was scrolled.

**CMA\_HORIZONTAL** A window was scrolled horizontally. If the split details view window is scrolled, a logical OR operator (|) is used to combine the CMA\_HORIZONTAL attribute with either the CMA\_LEFT

attribute or the CMA\_RIGHT attribute to indicate which window was scrolled. If the unsplit details view window is scrolled, the CMA\_HORIZONTAL attribute is combined with the CMA\_LEFT attribute.

CMA\_VERTICAL

The container window scrolled vertically. If the split details view window is scrolled, a logical OR operator (|) is used to combine the CMA\_VERTICAL attribute with the CMA\_LEFT attribute and the CMA\_RIGHT attribute. If the unsplit details view window is scrolled, the CMA\_VERTICAL attribute is combined with the CMA\_LEFT attribute.

---

## OBJCLASS

Object class structure.

### Syntax

```
typedef struct _OBJCLASS {
    struct _OBJCLASS *pNext;
    PSZ                pszClassName;
    PSZ                pszModName;
} OBJCLASS;

typedef OBJCLASS *POBJCLASS;
```

### Fields

**pNext** (struct \_OBJCLASS \*)

Pointer to the next object class structure.

**pszClassName** (PSZ)

Class name.

**pszModName** (PSZ)

Module name.

---

## OWNERBACKGROUND

Structure that contains information about painting the container window's background by the container owner. This structure is used in the CM\_PAINTBACKGROUND container message only. See "CM\_PAINTBACKGROUND" on page 22-55 for information about that message.

### Syntax

```
typedef struct _OWNERBACKGROUND {
    HWND      hwnd;
    HPS       hps;
    RECTL     rc1Background;
    LONG      idWindow;
} OWNERBACKGROUND;

typedef OWNERBACKGROUND *POWNERBACKGROUND;
```

### Fields

#### **hwnd** (HWND)

Window handle.

Handle of the window to be painted.

#### **hps** (HPS)

Presentation-space handle.

#### **rc1Background** (RECTL)

Background rectangle.

Background rectangle in window coordinates.

#### **idWindow** (LONG)

Window ID.

Identity of the window to be painted.

---

## OWNERITEM

Owner item.

### Syntax

```
typedef struct _OWNERITEM {
    HWND      hwnd;
    HPS       hps;
    ULONG     fsState;
    ULONG     fsAttribute;
    ULONG     fsStateOld;
    ULONG     fsAttributeOld;
    RECTL     rcItem;
    LONG      idItem;
    ULONG     hItem;
} OWNERITEM;

typedef OWNERITEM *POWNERITEM;
```

### Fields

**hwnd** (HWND)

Window handle.

**hps** (HPS)

Presentation-space handle.

**fsState** (ULONG)

State.

**fsAttribute** (ULONG)

Attribute.

**fsStateOld** (ULONG)

Old state.

**fsAttributeOld** (ULONG)

Old attribute.

**rcItem** (RECTL)

Item rectangle.

**idItem** (LONG)

Item identity.

**hItem** (ULONG)

Item.

---

## **MLEOVERFLOW**

Overflow error structure for multiline entry field.

### **Syntax**

```
typedef struct MLEOVERFLOW {  
    ULONG    afErrInd;  
    LONG     nBytesOver;  
    LONG     pixHorzOver;  
    LONG     pixVertOver;  
} MLEOVERFLOW;  
  
typedef MLEOVERFLOW *POVERFLOW;
```

### **Fields**

**afErrInd** (ULONG)

One or more EFR\_\* flags.

**nBytesOver** (LONG)

Number of bytes over the limit.

**pixHorzOver** (LONG)

Number of pels over the horizontal limit.

**pixVertOver** (LONG)

Number of pels over the vertical limit.



---

## PAGEINFO

Settings page information structure.

### Syntax

```
typedef struct _PAGEINFO {
    ULONG      cb;
    HWND      hwndPage;
    PFNWP     pfnwp;
    ULONG     resid;
    PVOID     pCreateParams;
    USHORT    dlgid;
    USHORT    usPageStyleFlags;
    USHORT    usPageInsertFlags;
    USHORT    usSettingsFlags;
    PSZ      pszName;
    USHORT    idDefaultHelpPanel;
    USHORT    usReserved2;
    PSZ      pszHelpLibraryName;
    PUSHORT  pHelpSubtable;
    HMODULE   hmodHelpSubtable;
    ULONG     ulPageInsertId;
} PAGEINFO;

typedef PAGEINFO *PPAGEINFO;
```

### Fields

**cb** (ULONG)

Length of PAGEINFO structure.

**hwndPage** (HWND)

Handle of page.

**pfnwp** (PFNWP)

Window procedure.

**resid** (ULONG)

Resource identity.

**pCreateParams** (PVOID)

Pointer to creation parameters.

**dlgid** (USHORT)

Dialog identity.

**usPageStyleFlags** (USHORT)

Notebook control page style flags.

**usPageInsertFlags** (USHORT)

Notebook control page insertion flags.

**usSettingsFlags** (USHORT)

Settings flag.

This flag must be set to one of the following values:

0 You will not get page numbers.

SETTINGS\_PAGE\_NUMBERS Page numbers will automatically be put on the status line for pages that have minor pages under the major tab page.

If you want to use the page numbers, make sure ALL pages have this setting.

**pszName** (PSZ)

Pointer to a string containing page name.

**idDefaultHelpPanel** (USHORT)

Identity of default help panel.

**usReserved2** (USHORT)

Reserved value, must be zero.

**pszHelpLibraryName** (PSZ)

Pointer to name of help file.

**pHelpSubtable** (PUSHORT)

Pointer to help subtable.

**hmodHelpSubtable** (HMODULE)

Module handle for help subtable.

**ulPageInsertid** (ULONG)

Notebook control page identity.

---

## PAGESELECTNOTIFY

Structure that contains information about the application page being selected.

### Syntax

```
typedef struct _PAGESELECTNOTIFY {
    HWND      hwndBook;
    ULONG     ulPageIdCur;
    ULONG     ulPageIdNew;
} PAGESELECTNOTIFY;

typedef PAGESELECTNOTIFY *PPAGESELECTNOTIFY;
```

### Fields

#### hwndBook (HWND)

Notebook window handle.

#### ulPageIdCur (ULONG)

Current top page identifier.

#### ulPageIdNew (ULONG)

New top page identifier.

---

## PANOSE

The *Panose* field in the font metrics will allow for quantitative descriptions of the visual properties of font faces. The PANOSE definition contains ten digits, each of which currently describes up to sixteen variations.

### Syntax

```
typedef struct _PANOSE {
    BYTE      bFamilyType;
    BYTE      bSerifStyle;
    BYTE      bWeight;
    BYTE      bProportion;
    BYTE      bContrast;
    BYTE      bStrokeVariation;
    BYTE      bArmStyle;
    BYTE      bLetterform;
    BYTE      bMidline;
    BYTE      bXHeight;
    BYTE      fbPassedISO;
    BYTE      fbFailedISO;
} PANOSE;

typedef PANOSE *PPANOSE;
```

## **Fields**

### **bFamilyType (BYTE)**

Family kind.

- 0 Any
- 1 No Fit
- 2 Text and Display
- 3 Script
- 4 Decorative
- 5 Pictorial

### **bSerifStyle (BYTE)**

Serif style.

- 0 Any
- 1 No Fit
- 2 Cove
- 3 Obtuse Cove
- 4 Square Cove
- 5 Obtuse Square Cove
- 6 Square
- 7 Thin
- 8 Bone
- 9 Exaggerated
- 10 Triangle
- 11 Normal Sans
- 12 Obtuse Sans
- 13 Perp Sans
- 14 Flared
- 15 Rounded

### **bWeight (BYTE)**

Weight.

- 0 Any
- 1 No Fit
- 2 Very Light
- 3 Light
- 4 Thin
- 5 Book
- 6 Medium
- 7 Demi
- 8 Bold
- 9 Heavy
- 10 Black
- 11 Nord

**bProportion (BYTE)**

Proportion.

- 0 Any
- 1 No Fit
- 2 Old Style
- 3 Modern
- 4 Even Width
- 5 Expanded
- 6 Condensed
- 7 Very Expanded
- 8 Very Condensed
- 9 Monospaced

**bContrast (BYTE)**

Contrast.

- 0 Any
- 1 No Fit
- 2 None
- 3 Very Low
- 4 Low
- 5 Medium Low
- 6 Medium
- 7 Medium High
- 8 High
- 9 Very High

**bStrokeVariation (BYTE)**

Stroke Variation.

- 0 Any
- 1 No Fit
- 2 Gradual/Diagonal
- 3 Gradual/Transitional
- 4 Gradual/Vertical
- 5 Gradual/Horizontal
- 6 Rapid/Vertical
- 7 Rapid/Horizontal
- 8 Instant/Vertical

**bArmStyle (BYTE)**

Arm Style.

- 0 Any
- 1 No Fit
- 2 Straight Arms/Horizontal
- 3 Straight Arms/Wedge
- 4 Straight Arms/Vertical
- 5 Straight Arms/Single Serif
- 6 Straight Arms/Double Serif

- 7 Non-Straight Arms/Horizontal
- 8 Non-Straight Arms/Wedge
- 9 Non-Straight Arms/Vertical
- 10 Non-Straight Arms/Single Serif
- 11 Non-Straight Arms/Double Serif

**bLetterform (BYTE)**

Letterform.

- 0 Any
- 1 No Fit
- 2 Normal/Contact
- 3 ONormal/Weighted
- 4 ONormal/Boxed
- 5 ONormal/Flattened
- 6 ONormal/Rounded
- 7 ONormal/Off Center
- 8 ONormal/Square
- 9 Oblique/Contact
- 10 Oblique/Weighted
- 11 Oblique/Boxed
- 12 Oblique/Flattened
- 13 Oblique/Rounded
- 14 Oblique/Off Center
- 15 Oblique/Square

**bMidline (BYTE)**

Midline.

- 0 Any
- 1 No Fit
- 2 Standard/Trimmed
- 3 Standard/Pointed
- 4 Standard/Serifed
- 5 High/Trimmed
- 6 High/Pointed
- 7 High/Serifed
- 8 Constant/Trimmed
- 9 Constant/Pointed
- 10 Constant/Serifed
- 11 Low/Trimmed
- 12 Low/Pointed
- 13 Low/Serifed

**bXHeight (BYTE)**

X-Height.

- 0 Any
- 1 No Fit
- 2 Constant/Small
- 3 Constant/Standard
- 4 Constant/Large
- 5 Ducking/Small
- 6 Ducking/Standard
- 7 Ducking/Large

**fbPassedISO (BYTE)**

Font passed ISO test.

The following flags indicate those displays and resolutions at which the font complied with ISO 9241.

```
FM_ISO_9518_640
FM_ISO_9515_640
FM_ISO_9515_1024
FM_ISO_9517_640
FM_ISO_9517_1024
```

**fbFailedISO (BYTE)**

Font failed ISO test.

The following flags indicate those displays and resolutions at which the font did not comply with ISO 9241.

```
FM_ISO_9518_640
FM_ISO_9515_640
FM_ISO_9515_1024
FM_ISO_9517_640
FM_ISO_9517_1024
```

---

**PARAM**

Presentation parameter attribute definition.

**Syntax**

```
typedef struct _PARAM {
    ULONG    id;
    ULONG    cb;
    BYTE     ab[1];
} PARAM;

typedef PARAM *PPARAM;
```

## Fields

### id (ULONG)

Attribute type identity.

These identities are in the range of 0x00000000 to 0xFFFFFFFF. The window manager uses values of this parameter in the range 0x00000000 to PP\_USER; therefore, an application cannot define private presentation parameter attribute identities in this range. An application should use WinAddAtom to guarantee obtaining a unique identity.

|                                       |   |
|---------------------------------------|---|
| PP_FOREGROUND_COLOR                   | Foreground color (in RGB) attribute.  |
| PP_BACKGROUND_COLOR                   | Background color (in RGB) attribute.  |
| PP_FOREGROUND_COLOR_INDEX             | Foreground color index attribute.   |
| PP_BACKGROUND_COLOR_INDEX             | Background color index attribute.   |
| PP_HILITE_FOREGROUND_COLOR            | Highlighted foreground color (in RGB) attribute, for example for selected menu items. |
| PP_HILITE_BACKGROUND_COLOR            | Highlighted background color (in RGB) attribute.                                      |
| PP_HILITE_FOREGROUND_COLOR_INDEX      | Highlighted foreground color index attribute.   |
| PP_HILITE_BACKGROUND_COLOR_INDEX      | Highlighted background color index attribute.   |
| PP_DISABLED_FOREGROUND_COLOR          | Disabled foreground color (in RGB) attribute.   |
| PP_DISABLED_BACKGROUND_COLOR          | Disabled background color (in RGB) attribute.   |
| PP_DISABLED_FOREGROUND_COLOR_INDEX    | Disabled foreground color index attribute.  |
| PP_DISABLED_BACKGROUND_COLOR_INDEX    | Disabled background color index attribute.  |
| PP_BORDER_COLOR                       | Border color (in RGB) attribute.  |
| PP_BORDER_COLOR_INDEX                 | Border color index attribute.   |
| PP_FONT_NAME_SIZE                     | Font name and size attribute.   |
| PP_ACTIVE_COLOR                       | Active color value of data type RGB.  |
| PP_ACTIVE_COLOR_INDEX                 | Active color index value of data type LONG.   |
| PP_INACTIVE_COLOR                     | Inactive color value of data type RGB.  |
| PP_INACTIVE_COLOR_INDEX               | Inactive color index value of data type LONG.   |
| PP_ACTIVE_TEXT_FOREGROUND_COLOR       | Active text foreground color value of data type RGB.                                  |
| PP_ACTIVE_TEXT_FOREGROUND_COLOR_INDEX | Active text foreground color index value of data type LONG.                           |
| PP_ACTIVE_TEXT_BACKGROUND_COLOR       | Active text background color value of data type RGB.                                  |
| PP_ACTIVE_TEXT_BACKGROUND_COLOR_INDEX | Active text background color index value of data type LONG.                           |
| PP_INACTIVE_TEXT_FOREGROUND_COLOR     | Inactive text foreground color value of data type RGB.                                |



|                               |   |
|-------------------------------|---|
| PP_INACTIVETEXTFGNDCOLORINDEX | Inactive text foreground color index value of data type LONG. |
| PP_INACTIVETEXTBGNDCOLOR      | Inactive text background color value of data type RGB.        |
| PP_INACTIVETEXTBGNDCOLORINDEX | Inactive text background color index value of data type LONG. |
| PP_SHADOW                     | Changes the color used for drop shadows on certain controls.  |
| PP_USER                       | This is a user-defined presentation parameter.                |

**cb (ULONG)**

Byte count of the *ab* parameter.

**ab[1] (BYTE)**

Attribute value.

The format of a value depends on the attribute type identity as follows:

|                                  |   |
|----------------------------------|---|
| PP_FOREGROUNDCOLOR               | Foreground color value of data type RGB.  |
| PP_BACKGROUNDDCOLOR              | Background color value of data type RGB.  |
| PP_FOREGROUNDCOLORINDEX          | Foreground color index value of data type LONG.   |
| PP_BACKGROUNDDCOLORINDEX         | Background color index value of data type LONG.   |
| PP_HILITEFOREGROUNDCOLOR         | Highlighted foreground color value of data type RGB.  |
| PP_HILITEBACKGROUNDDCOLOR        | Highlighted background color value of data type RGB.  |
| PP_HILITEFOREGROUNDCOLORINDEX    | Highlighted foreground color index value of data type LONG.   |
| PP_HILITEBACKGROUNDDCOLORINDEX   | Highlighted background color index value of data type LONG.   |
| PP_DISABLEDFOREGROUNDCOLOR       | Disabled foreground color value of data type RGB.   |
| PP_DISABLEDBACKGROUNDDCOLOR      | Disabled background color value of data type RGB.   |
| PP_DISABLEDFOREGROUNDCOLORINDEX  | Disabled foreground color index value of data type LONG.  |
| PP_DISABLEDBACKGROUNDDCOLORINDEX | Disabled background color index value of data type LONG.  |
| PP_BORDERCOLOR                   | Border color value of data type RGB.  |
| PP_BORDERCOLORINDEX              | Border color index value of data type LONG.   |
| PP_FONTNAMESIZE                  | Font name and size values, of data type PSZ. The string is in two parts, separated by a period. The first part is the font point size and the second part |

is the font facename, for example,  
"12.Helv."

---

## **PCH**

Pointer to a character string.

### **Syntax**

```
typedef unsigned char * PCH;
```

---

## **PCSZ**

Pointer to a constant null-terminated string.

### **Syntax**

```
typedef const char * PCSZ;
```

---

## **PDEVOPENDATA**

Open device-data array.

This data type points to data whose format is described by the DEVOPENSTRUC data type.

### **Syntax**

```
typedef PSZ * PDEVOPENDATA;
```

---

## PFN

Pointer to a procedure.

### Syntax

```
typedef _PFN *PFN;
```

In the header file, this is a two-part definition as shown below:

```
typedef int (APIENTRY _PFN) ();  
typedef _PFN *PFN;
```

---

## PFNWP

Pointer to a window procedure.

This is the standard function definition for window procedures.

### Syntax

```
typedef FWP *PFNWP;
```

The first argument (HWND) is the handle of the window receiving the message. The second argument (ULONG) is a message identifier. The third argument (MPARAM) is the first message parameter (mp1). The fourth argument (MPARAM) is the second message parameter (mp2). The function returns an HRESULT. Each message has a specific set of possible return codes. The window procedure must return a value that is appropriate for the message being processed.

In the header file, this is a two-part definition as shown below:

```
typedef HRESULT (EXFENTRY FWP)(HWND, ULONG, MPARAM, MPARAM);  
typedef FWP *PFNWP;
```

Window procedures must be EXPORTED in the definitions file used by the linker.

---

## PID

Process identity.

### Syntax

```
typedef LHANDLE PID;
```

---

## PIX

Pel count for multi-line entry field.

### Syntax

```
typedef LONG PIX;
```

---

## PRDINFO3

Print device information structure (level 3).

### Syntax

```
typedef struct _PRDINFO3 {  
    PSZ      pszPrinterName;  
    PSZ      pszUserName;  
    PSZ      pszLogAddr;  
    USHORT   uJobId;  
    USHORT   fsStatus;  
    PSZ      pszStatus;  
    PSZ      pszComment;  
    PSZ      pszDrivers;  
    USHORT   time;  
    USHORT   usTimeOut;  
} PRDINFO3;  
  
typedef PRDINFO3 *PPRDINFO3;
```

### Fields

#### **pszPrinterName** (PSZ)

Print device name.

#### **pszUserName** (PSZ)

User who submitted job.

This parameter is valid only while the job is printing. It is NULL for a job submitted locally.

#### **pszLogAddr** (PSZ)

Logical address (for example LPT1).

If NULL or an empty string, the printer is not connected to a logical address.

#### **uJobId** (USHORT)

Identity of current job.

If 0, no job is printing.

**fsStatus** (USHORT)

Print destination status.

Use the mask PRD\_STATUS\_MASK to determine the print job status:

|            |                            |
|------------|----------------------------|
| PRD_ACTIVE | Processing                 |
| PRD_PAUSED | Not processing, or paused. |

Use the mask PRJ\_DEVSTATUS for further information about print job status:

|                 |   |
|-----------------|---|
| PRJ_COMPLETE    | Job complete  |
| PRJ_INTERV      | Intervention required   |
| PRJ_ERROR       | Error occurred (in this case, <i>pszStatus</i> may contain a comment about the error) |
| PRJ_DESTOFFLINE | Print device offline  |
| PRJ_DESTPAUSED  | Print device paused   |
| PRJ_NOTIFY      | Raise alert   |
| PRJ_DESTNOPAPER | Print device out of paper.  |

**pszStatus** (PSZ)

Print device comment while printing.

A comment posted by the print processor of the print device. This parameter is valid only during printing.

**pszComment** (PSZ)

Print device description.

**pszDrivers** (PSZ)

Drivers supported by print device.

List items are separated by commas. Each printer driver name may have a device name separated by a dot (for example, PLOTTER.HP7475A). The default printer is listed first.

**time** (USHORT)

Time job has been printing (minutes)

This parameter applies only during printing.

**usTimeOut** (USHORT)

Device timeout (seconds)

The time that elapses before the device driver notifies the spooler that the print device has not responded.

---

## PRDRIVINFO

Printer driver information structure (level 0).

### Syntax

```
typedef struct _PRDRIVINFO {  
    CHAR        szDrvName[DRIV_NAME_SIZE+1+DRIV_DEVICENAME_SIZE+1];  
} PRDRIVINFO;  
  
typedef PRDRIVINFO *PPRDRIVINFO;
```

### Fields

**szDrvName[DRIV\_NAME\_SIZE+1+DRIV\_DEVICENAME\_SIZE+1]** (CHAR)

Name of printer driver.

This is the name of the printer driver and device is the format of DRIVER.DEVICE. For example "IBM4019.IBM Laserprinter E."

---

## PRESPARAMS

Presentation parameter data.

### Syntax

```
typedef struct _PRESPARAMS {  
    ULONG        cb;  
    PARAM        aparam[1];  
} PRESPARAMS;  
  
typedef PRESPARAMS *PPRESPARAMS;
```

### Fields

**cb** (ULONG)

Length of the *aparam* parameter, in bytes.

**aparam[1]** (PARAM)

Array of presentation attribute parameters.

---

## PRINTDEST

PRINTDEST data structure.

Contains all the parameters required to issue a `DevPostDeviceModes` and `DevOpenDC` function calls.

### Syntax

```
typedef struct _PRINTDEST {
    ULONG          cb;
    LONG           IType;
    PSZ            pszToken;
    LONG           ICount;
    PDEVOPENDATA  pdopData;
    ULONG          fl;
    PSZ            pszPrinter;
} PRINTDEST;

typedef PRINTDEST *PPRINTDEST;
```

### Fields

#### **cb** (ULONG)

Length of data structure, in bytes.

#### **IType** (LONG)

Type of device context.

`OD_QUEUED` The device context is queued.

`OD_DIRECT` The device context is direct.

#### **pszToken** (PSZ)

Device-information token.

This is always "\*".

#### **ICount** (LONG)

Number of items.

This is the number of items present in the *pdopData* field.

#### **pdopData** (PDEVOPENDATA)

Open device context data area.

See `DEVOPENSTRUC` for information on the format of *pdopData*.

#### **fl** (ULONG)

Flags.

`PD_JOB_PROPERTY` This flag indicates that `DevPostDeviceModes` should be called with `DPDM_POSTJOBPROP` before calling `DevOpenDC`.

**pszPrinter (PSZ)**

Name of Printer name.

A name that specifies the device, for example "PRINTER1." The name is used for calling DevPostDeviceModes.

The printer device name can be found by calling SplQueryQueue and passing to it the information found in the *pszLogAddress* field of the DEVOPENSTRUC structure pointed to by *pdopData*. SplQueryQueue returns a PRQINFO3 structure. The *pszPrinters* field in PRQINFO3 contains the printer device name to be used.

**PRINTERINFO**

Print destination information structure.

This structure is used at information level 0.

**Syntax**

```
typedef struct _PRINTERINFO {
    ULONG      flType;
    PSZ        pszComputerName;
    PSZ        pszPrintDestinationName;
    PSZ        pszDescription;
    PSZ        pszLocalName;
} PRINTERINFO;

typedef PRINTERINFO *PPRINTERINFO;
```

**Fields****flType (ULONG)**

Type of printer.

This is a flag used to describe the type of print destination:

|                      |  |
|----------------------|--|
| SPL_PR_QUEUE         | Print destination is a queue               |
| SPL_PR_DIRECT_DEVICE | Print destination is a direct print device |
| SPL_PR_QUEUED_DEVICE | Print destination is a queued print device |

**pszComputerName (PSZ)**

Computer name.

A NULL string specifies the local workstation.

**pszPrintDestinationName (PSZ)**

Name of Print Destination.

This is either a queue name or a print device name depending upon the value of *flType*. The maximum length of the name in the network case is 256 (including one byte for the null terminator).



**pszDescription (PSZ)**

Description of print destination.

The maximum length is 48 characters (including one byte for the null terminator).

**pszLocalName (PSZ)**

Local name of remote print destination.

This is a local port name (for instance "LPT4") that is connected to the remote print destination. A NULL string specifies that no connection exists.

---

**PRFPROFILE**

Profile structure.

**Syntax**

```
typedef struct _PRFPROFILE {
    ULONG      cchUserName;
    PSZ        pszUserName;
    ULONG      cchSysLen;
    PSZ        pszSysName;
} PRFPROFILE;

typedef PRFPROFILE *PPRFPROFILE;
```

**Fields****cchUserName (ULONG)**

Length of user profile name.

**pszUserName (PSZ)**

User profile name.

**cchSysLen (ULONG)**

Length of system profile name.

**pszSysName (PSZ)**

System profile name.

---

## PRJINFO2

Print-job information structure.

This structure provides a subset of the information supplied by PRJINFO3. It minimizes the storage required for job-information retrieval, and is sufficient for most uses.

### Syntax

```
typedef struct _PRJINFO2 {
    USHORT    uJobId;
    USHORT    uPriority;
    PSZ       pszUserName;
    USHORT    uPosition;
    USHORT    fsStatus;
    ULONG     ulSubmitted;
    ULONG     ulSize;
    PSZ       pszComment;
    PSZ       pszDocument;
} PRJINFO2;

typedef PRJINFO2 *PPRJINFO2;
```

### Fields

#### **uJobId** (USHORT)

Job identification number.

#### **uPriority** (USHORT)

Job priority.

The job-priority range is 1 through 99, with 99 the highest job priority. (For queue priorities, 1 is the highest priority.)

The job priority determines the order of jobs in the queue. If multiple queues print to the same printer, the job at the front of each queue is examined. The job with the highest priority is printed first; if there is more than one job with the highest priority, the oldest job with this priority is printed first.

|                         |                  |
|-------------------------|------------------|
| <b>PRJ_MAX_PRIORITY</b> | Highest priority |
| <b>PRJ_MIN_PRIORITY</b> | Lowest priority  |
| <b>PRJ_NO_PRIORITY</b>  | No priority.     |

#### **pszUserName** (PSZ)

User who submitted the job.

This parameter applies only to jobs created by a user and enqueued on a remote server. A NULL string signifies a local job.

#### **uPosition** (USHORT)

Job position in queue.

If 1, the job is scheduled to be the next job printed from this queue.

**fsStatus** (USHORT)

Job status.

To find the job status, use the PRJ\_QSTATUS mask:

|                 |   |
|-----------------|---|
| PRJ_QS_QUEUED   | Queued                                  |
| PRJ_QS_PAUSED   | Paused by a SpilHoldJob function        |
| PRJ_QS_SPOOLING | Job being created                       |
| PRJ_QS_PRINTING | Printing (bits 2 through 11 are valid). |

For further information, use the PRJ\_DEVSTATUS mask:

|                 |                                      |
|-----------------|--------------------------------------|
| PRJ_COMPLETE    | Job complete                         |
| PRJ_INTERV      | Intervention required                |
| PRJ_ERROR       | Error occurred.                      |
| PRJ_DESTOFFLINE | Print destination offline            |
| PRJ_DESTPAUSED  | Print destination paused             |
| PRJ_NOTIFY      | Alert should be raised               |
| PRJ_DESTNOPAPER | Print destination out of paper       |
| PRJ_DESTFORMCHG | Printer waiting for form change      |
| PRJ_DESTCRTCHG  | Printer waiting for cartridge change |
| PRJ_DESTPENCHG  | Printer waiting for pen change.      |

This bit indicates that the job is deleted:

|             |              |
|-------------|--------------|
| PRJ_DELETED | Job deleted. |
|-------------|--------------|

**ulSubmitted** (ULONG)

Time job submitted.

Time format is the same as that stored in the global information segment.

**ulSize** (ULONG)

Print-job size (bytes).

**pszComment** (PSZ)

Comment string.

Information about the print job. The maximum length of the string is 48 characters (including one byte for the null terminator).

**pszDocument** (PSZ)

Document name.

The document name of the print job (set by the application that submitted the print job). The maximum length of the string is 260 characters.

## PRJINFO3

Print-job information structure.

This structure is used when complete job details are required. A subset of this information is supplied by PRJINFO2.

### Syntax

```
typedef struct _PRJINFO3 {
    USHORT      uJobId;
    USHORT      uPriority;
    PSZ         pszUserName;
    USHORT      uPosition;
    USHORT      fsStatus;
    ULONG       ulSubmitted;
    ULONG       ulSize;
    PSZ         pszComment;
    PSZ         pszDocument;
    PSZ         pszNotifyName;
    PSZ         pszDataType;
    PSZ         pszParms;
    PSZ         pszStatus;
    PSZ         pszQueue;
    PSZ         pszQProcName;
    PSZ         pszQProcParms;
    PSZ         pszDriverName;
    PDRIVDATA   pDriverData;
    PSZ         pszPrinterName;
} PRJINFO3;

typedef PRJINFO3 *PPRJINFO3;
```

### Fields

#### **uJobId** (USHORT)

Job identification number.

#### **uPriority** (USHORT)

Job priority.

The job-priority range is 1 through 99, with 99 the highest job priority. (For queue priorities, 1 is the highest priority.)

The job priority determines the order of jobs in the queue. If multiple queues print to the same printer, the job on the front of each queue is examined. The job with the highest priority is printed first; if there is more than one job with the highest priority, the oldest job with this priority is printed first.

|                  |                  |
|------------------|------------------|
| PRJ_MAX_PRIORITY | Highest priority |
| PRJ_MIN_PRIORITY | Lowest priority  |
| PRJ_NO_PRIORITY  | No priority.     |

**pszUserName** (PSZ)

User who submitted the job.

This parameter applies only to jobs created by a user on a remote workstation and queued on a server. A NULL string signifies a local job.

**uPosition** (USHORT)

Job position in queue.

If 1, the job is scheduled to be the next job printed from this queue.

**fsStatus** (USHORT)

Job status.

To find the job status, use the PRJ\_QSTATUS mask:

**ulSubmitted** (ULONG)

Time job submitted.

Time format is the same as that stored in the global information segment.

**ulSize** (ULONG)

Print-job size (bytes).

**pszComment** (PSZ)

Comment string.

Information about the print job.

The maximum length of the string is 48 characters (including one byte for the null terminator).

**pszDocument** (PSZ)

Document name.

The document name of the print job (set by the application that submitted the print job). The maximum length of the string is 260 characters.

**pszNotifyName** (PSZ)

Messaging alias for print alert.

This parameter is a computer name and applies only to jobs on a remote server queue. A NULL string is returned for jobs on a local queue.

**pszDataType** (PSZ)

Data type of submitted file.

This is specified by the *pszDataType* parameter in the DEVOPENSTRUC structure passed to the DevOpenDC call when the job is created. The name is truncated to fit the field if necessary, and contains a trailing NULL.

**pszParms** (PSZ)

Parameters.

The form of this string is:

parm1=val1 parm2=val2 ...

**pszStatus (PSZ)**

Status comment.

A text string, posted by the queue processor, that provides additional job-status information. The default string type is NULL.

**pszQueue (PSZ)**

Queue name.

The name of the queue the job is on.

**pszQProcName (PSZ)**

Queue processor.

The name of the queue processor.

**pszQProcParms (PSZ)**

Queue processor parameters.

Spaces are used to separate parameters.

**pszDriverName (PSZ)**

Driver name.

The name of the device driver (for example, "LASERJET"). The device name is part of *pDriverData*.

**pDriverData (PDRIVDATA)**

Job Properties (driver data).

The contents are specific to the device driver.

**pszPrinterName (PSZ)**

Printer name.

If the job is printing, the printer name, otherwise NULL.

---

**PROGRAMENTRY**

Program-entry structure.

**Syntax**

```
typedef struct _PROGRAMENTRY {
    HPROGRAM      hprog;
    PROGTYPE      progt;
    CHAR          szTitle[MAXNAMEL+1];
} PROGRAMENTRY;

typedef PROGRAMENTRY *PPROGRAMENTRY;
```

## Fields

**hprog** (HPROGRAM)  
Program handle.

**progt** (PROGTYPE)  
Program type.

**szTitle**[MAXNAMEL+1] (CHAR)  
Program title (null-terminated).

---

## PROGCATEGORY

Program category.

### Syntax

```
typedef ULONG PROGCATEGORY;
```

---

## PROGDETAILS

Program-details structure.

### Syntax

```
typedef struct _PROGDETAILS {  
    ULONG      Length;  
    PROGTYPE   progt;  
    PSZ        pszTitle;  
    PSZ        pszExecutable;  
    PSZ        pszParameters;  
    PSZ        pszStartupDir;  
    PSZ        pszIcon;  
    PSZ        pszEnvironment;  
    SWP        swpInitial;  
} PROGDETAILS;  
  
typedef PROGDETAILS *PPROGDETAILS;
```

## Fields

**Length** (ULONG)  
Length of structure.

**progt** (PROGTYPE)  
Program type.

**pszTitle (PSZ)**

Title.

**pszExecutable (PSZ)**

Executable file name.

**pszParameters (PSZ)**

Parameter string.

**pszStartupDir (PSZ)**

Start-up directory.

**pszIcon (PSZ)**

Icon-file name.

**pszEnvironment (PSZ)**

Environment string.

A list of null-terminated strings, ending with an extra NULL character.

**swpInitial (SWP)**

Initial window position and size.

---

**PROGTYPE**

Program-type structure.

**Syntax**

```
typedef struct _PROGTYPE {
    PROGCATEGORY    progc;
    ULONG           fbVisible;
} PROGTYPE;

typedef PROGTYPE *PPROGTYPE;
```

**Fields****progc (PROGCATEGORY)**

Program category:

PROG\_DEFAULT

Default application.

PROG\_PM

Presentation Manager application.

PROG\_WINDOWABLEVIO

Text-windowed application.

PROG\_FULLSCREEN

Full-screen application.

PROG\_WINDOWEDVDM

PC DOS executable process (windowed).

PROG\_VDM

PC DOS executable process (full screen).



|                           |   |
|---------------------------|---|
| PROG_REAL                 | PC DOS executable process (full screen). Same as PROG_VDM.  |
| PROG_31_STDSEAMLESSVDM    | Windows 3.1 program that will execute in its own windowed WINOS2 session.                                 |
| PROG_31_STDSEAMLESSCOMMON | Windows 3.1 program that will execute in a common windowed WINOS2 session.                                |
| PROG_31_ENHSEAMLESSVDM    | Windows 3.1 program that will execute in enhanced compatibility mode in its own windowed WINOS2 session.  |
| PROG_31_ENHSEAMLESSCOMMON | Windows 3.1 program that will execute in enhanced compatibility mode in a common windowed WINOS2 session. |
| PROG_31_ENH               | Windows 3.1 program that will execute in enhanced compatibility mode in a full screen WINOS2 session.     |
| PROG_31_STD               | Windows 3.1 program that will execute in a full screen WINOS2 session.                                    |

**fbVisible (ULONG)**

Visibility attribute.

When testing this field, allow for the possibility that other bits may be defined in the future. SHE\_INVISIBLE and SHE\_PROTECTED can be used to mask the visibility and protected flags, respectively.

|                 |             |
|-----------------|-------------|
| SHE_VISIBLE     | Visible     |
| SHE_INVISIBLE   | Invisible   |
| SHE_UNPROTECTED | Unprotected |
| SHE_PROTECTED   | Protected.  |

---

## PRPORTINFO

Port information structure (level 0).

### Syntax

```
typedef struct _PRPORTINFO {
    CHAR    szPortName[PDLLEN+1];
} PRPORTINFO;

typedef PRPORTINFO *PPRPORTINFO;
```

## Fields

**szPortName**[PDLEN+1] (CHAR)

Name of the port.

This is the name of the port. For example "LPT1."

---

## PRPORTINFO1

Port information structure (level 1).

## Syntax

```
typedef struct _PRPORTINFO1 {
    PSZ    pszPortName;
    PSZ    pszPortDriverName;
    PSZ    pszPortDriverPathName;
} PRPORTINFO1;

typedef PRPORTINFO1 *PPRPORTINFO1;
```

## Fields

**pszPortName** (PSZ)

Name of the port.

This is the name of the port. For example "LPT1."

**pszPortDriverName** (PSZ)

Name of the port driver.

This is the name of the port driver. For example "PARALLEL."

**pszPortDriverPathName** (PSZ)

Full path name of the port driver.

This is the full path name of the port driver. For example  
"C:\OS2\DLL\PARALLEL.PDR."

---

## PRQINFO3

Print-queue information structure.

This structure is used at information levels 3 and 4.

### Syntax

```
typedef struct _PRQINFO3 {
    PSZ          pszName;
    USHORT      uPriority;
    USHORT      uStartTime;
    USHORT      uUntilTime;
    USHORT      fsType;
    PSZ          pszSepFile;
    PSZ          pszPrProc;
    PSZ          pszParms;
    PSZ          pszComment;
    USHORT      fsStatus;
    USHORT      cJobs;
    PSZ          pszPrinters;
    PSZ          pszDriverName;
    PDRIVDATA   pDriverData;
} PRQINFO3;

typedef PRQINFO3 *PPRQINFO3;
```

### Fields

#### **pszName** (PSZ)

Queue name.

The maximum length of the name in the network case is 256 (including one byte for zero termination).

#### **uPriority** (USHORT)

Queue priority.

The range is 1 through 9, with 1 being the highest queue priority.

The default job priority (DefJobPrio) is determined from:  
DefJobPrio=100-(10\* uPriority).

If a job is added with *PRJ\_NO\_PRIORITY* specified, *DefJobPrio* is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

|                         |                  |
|-------------------------|------------------|
| <i>PRQ_DEF_PRIORITY</i> | Default priority |
| <i>PRQ_MAX_PRIORITY</i> | Highest priority |
| <i>PRQ_MIN_PRIORITY</i> | Minimum priority |
| <i>PRQ_NO_PRIORITY</i>  | No priority.     |

**uStartTime (USHORT)**

Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If *uStartTime* and *uUntilTime* are both 0, the print queue is always available.

**uUntilTime (USHORT)**

Minutes after midnight. when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If *uUntilTime* and *uStartTime* are both 0, the print queue is always available.

**fsType (USHORT)**

Queue type.

|                             |  |
|-----------------------------|--|
| <i>PRQ3_TYPE_RAW</i>        | Data is always enqueued in the device specific format.   |
| <i>PRQ3_TYPE_BYPASS</i>     | Allows the spooler to bypass the queue processor and send data directly to the Printer Driver. Setting this bit allows the spooler to print jobs of type <i>PM_Q_RAW</i> while they are still being spooled. |
| <i>PRQ3_TYPE_APPDEFAULT</i> | This bit is set for the application default queue only.  |

**pszSepFile (PSZ)**

Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

**pszPrProc (PSZ)**

Default queue-processor.

**pszParms (PSZ)**

Queue parameters.

This can be any text string or a NULL string.

**pszComment (PSZ)**

Queue description.

A NULL string results in no comment. The maximum length is 48 characters (including one byte for the null terminator).

**fsStatus** (USHORT)

Queue status.

PRQ3\_PAUSED     Queue is paused (held).  
PRQ3\_PENDING    Queue is pending deletion.

**cJobs** (USHORT)

Number of jobs in queue.

**pszPrinters** (PSZ)

Print devices connected to queue.

This cannot be NULL.

**pszDriverName** (PSZ)

Default device driver.

**pDriverData** (PDRIVDATA)

Default queue job properties.

**Note:** An application can use *pszDriverName*, *pDriverData*, *pszPrProc*, and *pszParms* to construct a valid DevOpenDC call based only on the queue name.

---

**PRQINFO6**

Print-queue information structure.

This structure is used at information level 6.

**Syntax**

```
typedef struct _PRQINFO6 {  
    PSZ          pszName;  
    USHORT      uPriority;  
    USHORT      uStartTime;  
    USHORT      uUntilTime;  
    USHORT      fsType;  
    PSZ          pszSepFile;  
    PSZ          pszPrProc;  
    PSZ          pszParms;  
    PSZ          pszComment;  
    USHORT      fsStatus;  
    USHORT      cJobs;  
    PSZ          pszPrinters;  
    PSZ          pszDriverName;  
    PDRIVDATA    pDriverData;  
    PSZ          pszRemoteComputerName;  
    PSZ          pszRemoteQueueName;  
} PRQINFO6;  
  
typedef PRQINFO6 *PPRQINFO6;
```

## Fields

### **pszName** (PSZ)

Queue name.

The maximum length of the name in the network case is 256 (including one byte for zero termination).

### **uPriority** (USHORT)

Queue priority.

The range is 1 through 9, with 1 being the highest queue priority.

The default job priority (DefJobPrio) is determined from:

$\text{DefJobPrio} = 100 - (10 * \text{uPriority})$ .

If a job is added with *PRJ\_NO\_PRIORITY* specified, DefJobPrio is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

|                  |                  |
|------------------|------------------|
| PRQ_DEF_PRIORITY | Default priority |
| PRQ_MAX_PRIORITY | Highest priority |
| PRQ_MIN_PRIORITY | Minimum priority |
| PRQ_NO_PRIORITY  | No priority.     |

### **uStartTime** (USHORT)

Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If *uStartTime* and *uUntilTime* are both 0, the print queue is always available.

### **uUntilTime** (USHORT)

Minutes after midnight. when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If *uUntilTime* and *uStartTime* are both 0, the print queue is always available.

### **fsType** (USHORT)

Queue type.

|                      |   |
|----------------------|---|
| PRQ3_TYPE_RAW        | Data is always enqueued in the device specific format.  |
| PRQ3_TYPE_BYPASS     | Allows the spooler to bypass the queue processor and send data directly to the Printer Driver. Setting this bit allows the spooler to print jobs of type PM_Q_RAW while they are still being spooled. |
| PRQ3_TYPE_APPDEFAULT | This bit is set for the application default queue only.   |

**pszSepFile** (PSZ)

Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

**pszPrProc** (PSZ)

Default queue-processor.

**pszParms** (PSZ)

Queue parameters.

This can be any text string or a NULL string.

**pszComment** (PSZ)

Queue description.

A NULL string results in no comment. The maximum length is 48 characters (including one byte for the null terminator).

**fsStatus** (USHORT)

Queue status.

PRQ3\_PAUSED     Queue is paused (held).  
PRQ3\_PENDING    Queue is pending deletion.

**cJobs** (USHORT)

Number of jobs in queue.

**pszPrinters** (PSZ)

Print devices connected to queue.

This cannot be NULL.

**pszDriverName** (PSZ)

Default device driver.

**pDriverData** (PDRIVDATA)

Default queue job properties.

**Note:** An application can use *pszDriverName*, *pDriverData*, *pszPrProc*, and *pszParms* to construct a valid DevOpenDC call based only on the queue name.

**pszRemoteComputerName** (PSZ)

Remote computer name.

The computer name part of a remote queue for which this queue is a local alias.

**pszRemoteQueueName** (PSZ)

Remote queue name.

The queue name part of a remote queue for which this queue is a local alias.

---

## PRQPROCINFO

Queue processor information structure (level 0).

### Syntax

```
typedef struct _PRQPROCINFO {
    CHAR      szQProcName[QNLEN+1];
} PRQPROCINFO;

typedef PRQPROCINFO *PPRQPROCINFO;
```

### Fields

**szQProcName[QNLEN+1]** (CHAR)

Name of queue processor.

This is the name of the queue processor (driver). For example "PMPRINT."

---

## POINTERINFO

Pointer-information structure.

### Syntax

```
typedef struct _POINTERINFO {
    ULONG      fPointer;
    LONG       xHotSpot;
    LONG       yHotSpot;
    HBITMAP    hbmPointer;
    HBITMAP    hbmColor;
    HBITMAP    hbmMiniPointer;
    HBITMAP    hbmMiniColor;
} POINTERINFO;

typedef POINTERINFO *PPOINTERINFO;
```

### Fields

**fPointer** (ULONG)

Bit-map size indicator.

TRUE Pointer-sized bit map

FALSE Icon-sized bit map.

**xHotSpot** (LONG)

X-coordinate of action point.



**yHotSpot (LONG)**

Y-coordinate of action point.

**hbmPointer (HBITMAP)**

Bit-map handle of pointer.

**hbmColor (HBITMAP)**

Bit-map handle of color bit map.

**hbmMiniPointer (HBITMAP)**

Bit-map handle of a pointer to a mini bit map.

**hbmMiniColor (HBITMAP)**

Bit-map handle of mini color bit map.

---

**POINTL**

Point structure (long integers).

**Syntax**

```
typedef struct _POINTL {  
    LONG    x;  
    LONG    y;  
} POINTL;  
  
typedef POINTL *PPPOINTL;
```

**Fields****x (LONG)**

X-coordinate.

**y (LONG)**

Y-coordinate.

---

## POINTS

Point structure (short integers).

### Syntax

```
typedef struct _POINTS {  
    SHORT    x;  
    SHORT    y;  
} POINTS;  
  
typedef POINTS *PPOINTS;
```

### Fields

**x** (SHORT)  
X-coordinate.

**y** (SHORT)  
Y-coordinate.

---

## PQMOPENDATA

Open queue-manager data array.

This data type points to data whose format is described by the DEVOPENSTRUC data type.

### Syntax

```
typedef PSZ * PQMOPENDATA;
```

---

## PSZ

Pointer to a null-terminated string.

If you are using C++ \*\*, you may need to use PCSZ.

### Syntax

```
typedef unsigned char * PSZ;
```

---

## PWPOINT

Pointer to a WPOINT data structure.

### Syntax

```
#define PWPOINT PPOINTL
```

---

## PVOID

Pointer to a data type of undefined format.

### Syntax

```
typedef VOID * PVOID;
```

---

## QMSG

Message structure.

### Syntax

```
typedef struct QMSG {  
    HWND      hwnd;  
    ULONG     msg;  
    MPARAM    mp1;  
    MPARAM    mp2;  
    ULONG     time;  
    POINTL    pt1;  
    ULONG     reserved;  
} QMSG;  
  
typedef QMSG *PQMSG;
```

### Fields

#### hwnd (HWND)

Window handle.

#### msg (ULONG)

Message identity.

#### mp1 (MPARAM)

Parameter 1.

- mp2** (MPARAM)  
Parameter 2.
- time** (ULONG)  
Message time.
- ptl** (POINTL)  
Pointer position when message was generated.
- reserved** (ULONG)  
Reserved.

---

## QUERYRECFROMRECT

Structure that contains information about a container record that is bounded by a specified rectangle. This structure is used in the `CM_QUERYRECORDFROMRECT` container message only. See “`CM_QUERYRECORDFROMRECT`” on page 22-62 for information about that message.

### Syntax

```
typedef struct _QUERYRECFROMRECT {
    ULONG      cb;
    RECTL      rect;
    ULONG      fsSearch;
} QUERYRECFROMRECT;

typedef QUERYRECFROMRECT *PQUERYRECFROMRECT;
```

### Fields

- cb** (ULONG)  
Structure size.  
The size (in bytes) of the `QUERYRECFROMRECT` data structure.
- rect** (RECTL)  
Rectangle.  
The rectangle to query, in virtual coordinates relative to the container window origin. If the details view (`CV_DETAIL`) is displayed, the x-coordinates of the rectangle are ignored.
- fsSearch** (ULONG)  
Search control flags.  
One flag from each of the following groups can be specified:
- Search sensitivity:

### **CMA\_COMPLETE**

Returns the container records that are completely within the bounding rectangle.

### **CMA\_PARTIAL**

Returns the container records that are completely or partially within the bounding rectangle.

- Enumeration order:

### **CMA\_ITEMORDER**

Container records are enumerated in item order, lowest to highest.

### **CMA\_ZORDER**

Container records are enumerated by z-order, from top to bottom. This flag is valid for the icon view only.

---

## **QUERYRECORDRECT**

Structure that contains information about the rectangle of the specified container record, relative to the container window origin. This structure is used in the `CM_QUERYRECORDRECT` container message only. See “`CM_QUERYRECORDRECT`” on page 22-64 for information about that message.

### **Syntax**

```
typedef struct _QUERYRECORDRECT {
    ULONG          cb;
    RECORDCORE     pRecord;
    ULONG          fRightSplitWindow;
    ULONG          fsExtent;
} QUERYRECORDRECT;

typedef QUERYRECORDRECT *PQUERYRECORDRECT;
```

### **Fields**

#### **cb (ULONG)**

Structure size.

The size (in bytes) of the `QUERYRECORDRECT` structure.

#### **pRecord (RECORDCORE)**

Pointer.

Pointer to the specified `RECORDCORE` data structure.

**Note:** If the `CCS_MINIRECORDCORE` style bit is specified when a container is created, then `MINIRECORDCORE` should be used instead of `RECORDCORE` and `PMINIRECORDCORE` should be used instead of `PRECORDCORE` in all applicable data structures and messages.

**fRightSplitWindow** (ULONG)

Window flag.

Flag that specifies the right or left window in the split details view.

This flag is ignored if the view is not the split details view.

TRUE Right split window is returned.

FALSE Left split window is returned.

**fsExtent** (ULONG)

Rectangle flags.

Flags that specify the extent of the desired rectangle.

These flags can be combined by using a logical OR operator (|) to return the rectangle that bounds the icon, the expanded and collapsed icon or bit map, and the text.

CMA\_ICON Returns the icon rectangle.

CMA\_TEXT Returns the text rectangle.

CMA\_TREEICON Returns the rectangle of the expanded and collapsed icons or bit maps. This flag is valid for the tree icon and tree text views only.

---

**RECORDCORE**

Structure that contains information for records in a container control. This data structure is used if the CCS\_MINIRECORDCORE style bit is not specified when a container is created.

**Syntax**

```
typedef struct _RECORDCORE {
    ULONG          cb;
    ULONG          flRecordAttr;
    POINTL         ptlIcon;
    struct _RECORDCORE *preccNextRecord;
    PSZ            pszIcon;
    HPOINTER       hptrIcon;
    HPOINTER       hptrMiniIcon;
    HBITMAP        hbmBitmap;
    HBITMAP        hbmMiniBitmap;
    PTREEITEMDESC  pTreeItemDesc;
    PSZ            pszText;
    PSZ            pszName;
    PSZ            pszTree;
} RECORDCORE;

typedef RECORDCORE *PRECORDCORE;
```

**Fields****cb** (ULONG)

The size, in bytes, of the RECORDCORE structure.

**flRecordAttr** (ULONG)

Container record attributes.

This parameter can contain any or all of the following:

|                       |  |
|-----------------------|--|
| <b>CRA_COLLAPSED</b>  | Specifies that a record is collapsed.  |
| <b>CRA_CURSORED</b>   | Specifies that a record will be drawn with a selection cursor.                 |
| <b>CRA_DISABLED</b>   | Specifies that a record will be drawn with unavailable-state emphasis.         |
| <b>CRA_DROPONABLE</b> | Specifies that a record can be a target for direct manipulation.               |
| <b>CRA_EXPANDED</b>   | Specifies that a record is expanded.   |
| <b>CRA_FILTERED</b>   | Specifies that a record is filtered and, therefore, hidden from view.          |
| <b>CRA_INUSE</b>      | Specifies that a record will be drawn with in-use emphasis.                    |
| <b>CRA_PICKED</b>     | Specifies that the container record will be picked up as part of the drag set. |
| <b>CRA_SELECTED</b>   | Specifies that a record will be drawn with selected-state emphasis.            |
| <b>CRA_SOURCE</b>     | Specifies that a record will be drawn with source-menu emphasis.               |

**ptlIcon** (POINTL)

Position of a container record in the icon view.

**preccNextRecord** (struct \_RECORDCORE \*)

Pointer to the next linked record.

**pszIcon** (PSZ)

Text for the icon view (CV\_ICON).

**hptrIcon** (HPOINTER)

Icon that is displayed when the CV\_MINI style bit is not specified.

This field is used when the CA\_DRAWICON container attribute of the CNRINFO data structure is set.

**hptrMinIcon** (HPOINTER)

Icon that is displayed when the CV\_MINI style bit is specified.

This field is used when the CA\_DRAWICON container attribute of the CNRINFO data structure is set.

**hbmBitmap** (HBITMAP)

Bit map displayed when the CV\_MINI style bit is not specified.

This field is used when the CA\_DRAWBITMAP container attribute of the CNRINFO data structure is set.

**hbmMiniBitmap** (HBITMAP)

Bit map displayed when the CV\_MINI style bit is specified.

This field is used when the CA\_DRAWBITMAP container attribute of the CNRINFO data structure is set.

**pTreeItemDesc** (PTREEITEMDESC)

Pointer to a TREEITEMDESC structure.

The TREEITEMDESC structure contains the icons and bit maps used to represent the state of an expanded or collapsed parent item in the tree name view.

**pszText** (PSZ)

Text for the text view (CV\_TEXT).

**pszName** (PSZ)

Text for the name view (CV\_NAME).

**pszTree** (PSZ)

Text for the tree view (CV\_TREE).

---

## RECORDINSERT

Structure that contains information about the RECORDCORE structure or structures that are being inserted into a container. The RECORDINSERT structure is used in the CM\_INSERTRECORD container message only. See "CM\_INSERTRECORD" on page 22-45 for information about that message.

**Note:** If the CCS\_MINIRECORDCORE style bit is specified when a container is created, then MINIRECORDCORE should be used instead of RECORDCORE and PMINIRECORDCORE should be used instead of PRECORDCORE in all applicable data structures and messages.

### Syntax

```
typedef struct _RECORDINSERT {
    ULONG          cb;
    PRECORDCORE    pRecordOrder;
    PRECORDCORE    pRecordParent;
    ULONG          fInvalidRecord;
    ULONG          zOrder;
    ULONG          cRecordsInsert;
} RECORDINSERT;

typedef RECORDINSERT *PRECORDINSERT;
```

### Fields



**cb (ULONG)**

Structure size.

The size (in bytes) of the RECORDINSERT structure.

**pRecordOrder (RECORDCORE)**

Record order.

Orders the RECORDCORE structure or structures relative to other RECORDCORE structures in the container. The values can be:

**CMA\_FIRST** Places a RECORDCORE structure, or list of RECORDCORE structures, at the beginning of the list of structures.

**CMA\_END** Places a RECORDCORE structure, or list of RECORDCORE structures, at the end of the list of structures.

**Other** Pointer to a RECORDCORE structure that this structure, or list of structures, is to be inserted after.

**pRecordParent (RECORDCORE)**

Pointer to record parent.

Pointer to a RECORDCORE structure that is the parent of the record or records to be inserted. This field is used only with the CMA\_FIRST or CMA\_END attributes of the *pRecordOrder* field.

**flinvalidateRecord (ULONG)**

Update flag.

Flag that indicates an automatic display update after RECORDCORE structures are inserted.

**TRUE** The display is automatically updated after a RECORDCORE structure is inserted.

**FALSE** The application must send the CM\_INVALIDATERECORD message after a RECORDCORE structure is inserted.

**zOrder (ULONG)**

Record z-order.

Positions the RECORDCORE structure in z-order, relative to other records in the container. The values can be:

**CMA\_TOP** Places a RECORDCORE structure at the top of the z-order. This is the default value.

**CMA\_BOTTOM** Places a RECORDCORE structure at the bottom of the z-order.

**cRecordsInsert (ULONG)**

Number of root level structures.

The number of root level RECORDCORE structures to be inserted. The *cRecordsInsert* field value must be greater than 0.

---

## RECTL

Rectangle structure.

### Syntax

```
typedef struct _RECTL {  
    LONG    xLeft;  
    LONG    yBottom;  
    LONG    xRight;  
    LONG    yTop;  
} RECTL;  
  
typedef RECTL *PRECTL;
```

### Fields

#### **xLeft** (LONG)

X-coordinate of left-hand edge of rectangle.

#### **yBottom** (LONG)

Y-coordinate of bottom edge of rectangle.

#### **xRight** (LONG)

X-coordinate of right-hand edge of rectangle.

#### **yTop** (LONG)

Y-coordinate of top edge of rectangle.

---

## RENDERFILE

File-rendering structure.

### Syntax

```
typedef struct _RENDERFILE {  
    HWND    hwndDragFiles;  
    HSTR    hstrSource;  
    HSTR    hstrTarget;  
    USHORT  fMove;  
    USHORT  usRsvd;  
} RENDERFILE;  
  
typedef RENDERFILE *PRENDERFILE;
```

## Fields

### **hwndDragFiles** (HWND)

Conversation handle.

Created by DrgDragFiles.

### **hstrSource** (HSTR)

Handle to source file name.

### **hstrTarget** (HSTR)

Handle to target file name.

### **fMove** (USHORT)

Operation.

TRUE Move the file.

FALSE Copy the file.

### **usRsvd** (USHORT)

Reserved.

---

## RGB

RGB color value.

## Syntax

```
typedef struct _RGB {  
    BYTE    bBlue;  
    BYTE    bGreen;  
    BYTE    bRed;  
} RGB;  
  
typedef RGB *PRGB;
```

## Fields

### **bBlue** (BYTE)

Blue component of the color definition.

### **bGreen** (BYTE)

Green component of the color definition.

### **bRed** (BYTE)

Red component of the color definition.

---

## RGB2

RGB color value.

### Syntax

```
typedef struct _RGB2 {  
    BYTE    bBlue;  
    BYTE    bGreen;  
    BYTE    bRed;  
    BYTE    fcOptions;  
} RGB2;  
  
typedef RGB2 *PRGB2;
```

### Fields

#### **bBlue** (BYTE)

Blue component of the color definition.

#### **bGreen** (BYTE)

Green component of the color definition.

#### **bRed** (BYTE)

Red component of the color definition.

#### **fcOptions** (BYTE)

Entry options.

These can be ORed together if required:

**PC\_RESERVED** The color entry is reserved for animating color with the palette manager.

**PC\_EXPLICIT** The low-order word of the color table entry designates a physical palette slot. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

---

## RGNRECT

Region-rectangle structure.

### Syntax

```
typedef struct _RGNRECT {
    ULONG      ircStart;
    ULONG      crc;
    ULONG      crcReturned;
    ULONG      ulDirection;
} RGNRECT;

typedef RGNRECT *PRGNRECT;
```

### Fields

#### **ircStart** (ULONG)

Rectangle number from which to start enumerating.

Numbering starts from 1.

#### **crc** (ULONG)

Number of rectangles that can be returned.

This must be 1 or greater.

#### **crcReturned** (ULONG)

Number of rectangles returned.

A value of less than *crc* indicates that there are no more rectangles to enumerate.

#### **ulDirection** (ULONG)

Direction in which the returned rectangles are to be ordered.

This ordering uses the leading edge of a rectangle:

|                     |                              |
|---------------------|------------------------------|
| RECTDIR_LFRT_TOPBOT | Left-to-right, top-to-bottom |
| RECTDIR_RTLF_TOPBOT | Right-to-left, top-to-bottom |
| RECTDIR_LFRT_BOTTOP | Left-to-right, bottom-to-top |
| RECTDIR_RTLF_BOTTOP | Right-to-left, bottom-to-top |

---

## SBCDATA

Scroll-bar control data structure.

## Syntax

```
typedef struct _SBCDATA {
    USHORT      cb;
    USHORT      sHilite;
    SHORT       posFirst;
    SHORT       posLast;
    SHORT       posThumb;
    SHORT       cVisible;
    SHORT       cTotal;
} SBCDATA;

typedef SBCDATA *PSBCDATA;
```

## Fields

### **cb** (USHORT)

Length of control data in bytes.

The length of the control data for a scroll-bar control.

This indicates which part of the scroll bar is to be highlighted, if any.

### **sHilite** (USHORT)

Highlighting code.

|                |                  |
|----------------|------------------|
| ZERO           | No highlighting  |
| SB_LINEUP      | Line up arrow    |
| SB_LINELEFT    | Line left arrow  |
| SB_LINEDOWN    | Line down arrow  |
| SB_LINERIGHT   | Line right arrow |
| SB_PAGEUP      | Page up arrow    |
| SB_PAGELLEFT   | Page left arrow  |
| SB_PAGEDOWN    | Page down arrow  |
| SB_PAGERIGHT   | Page right arrow |
| SB_SLIDERTRACK | Slider.          |

### **posFirst** (SHORT)

First bound of the scroll-bar range.

### **posLast** (SHORT)

Last bound of the scroll-bar range.

### **posThumb** (SHORT)

Slider position.

### **cVisible** (SHORT)

Number of data items visible.

### **cTotal** (SHORT)

Number of data items available.

---

## SEARCHSTRING

Structure that contains information about the container text string that is the object of the search. This structure is used in the CM\_SEARCHSTRING container message only. See "CM\_SEARCHSTRING" on page 22-71 for information about that message.

### Syntax

```
typedef struct _SEARCHSTRING {
    ULONG      cb;
    PSZ        pszSearch;
    ULONG      fsPrefix;
    ULONG      fsCaseSensitive;
    ULONG      usView;
} SEARCHSTRING;

typedef SEARCHSTRING *PSEARCHSTRING;
```

### Fields

#### **cb** (ULONG)

Structure size.

The size (in bytes) of the SEARCHSTRING structure.

#### **pszSearch** (PSZ)

Pointer to the search string.

#### **fsPrefix** (ULONG)

Search flag.

Search flag that defines the criteria by which the string specified by the *pszSearch* field is to be compared with the text of the container records to determine the pointer to the first matching record.

**TRUE** Matching occurs if the leading characters of the container record are the characters specified by the *pszSearch* field.

**FALSE** Matching occurs if the container record contains a substring of the characters specified by the *pszSearch* field.

#### **fsCaseSensitive** (ULONG)

Case sensitivity flag.

Determines case sensitivity of the search.

**TRUE** The search is case sensitive.

**FALSE** The search is not case sensitive.

## **usView (ULONG)**

View to search.

Search one of the container views for the string. Valid values are:

- CV\_ICON
- CV\_NAME
- CV\_TEXT
- CV\_TREE
- CV\_DETAIL

---

## **SEGOFF**

2-byte segment offset in bytes.

### **Syntax**

```
typedef follow SEGOFF;
```

---

## **SFACTORS**

Scaling factors. See DevEscape.

### **Syntax**

```
typedef struct _SFACTORS {  
    LONG    x;  
    LONG    y;  
} SFACTORS;  
  
typedef SFACTORS *PSFACTORS;
```

### **Fields**

**x (LONG)**

X-scaling factor, as an exponent of 2.

**y (LONG)**

Y-scaling factor, as an exponent of 2.



---

## SHORT

Signed integer in the range -32 768 through 32 767.

### Syntax

```
#define SHORT short
```

---

## SIZEF

Size structure (FIXED values).

### Syntax

```
typedef struct _SIZEF {  
    FIXED    cx;  
    FIXED    cy;  
} SIZEF;  
  
typedef SIZEF *PSIZEF;
```

### Fields

**cx** (FIXED)  
Width.

**cy** (FIXED)  
Height.

---

## SIZEL

Size structure (LONG values).

### Syntax

```
typedef struct _SIZEL {  
    LONG    cx;  
    LONG    cy;  
} SIZEL;  
  
typedef SIZEL *PSIZEL;
```

## Fields

**cx** (LONG)  
Width.

**cy** (LONG)  
Height.

---

## SLDCDATA

Slider control data structure.

## Syntax

```
typedef struct _SLDCDATA {
    ULONG      cbSize;
    USHORT     usScale1Increments;
    USHORT     usScale1Spacing;
    USHORT     usScale2Increments;
    USHORT     usScale2Spacing;
} SLDCDATA;

typedef SLDCDATA *PSLDCDATA;
```

## Fields

**cbSize** (ULONG)  
Data length.

Length of the control data in bytes.

**usScale1Increments** (USHORT)  
Scale increments.

The number of increments to set for the slider control. This number represents the range of values that can be selected within the slider when the SLS\_PRIMARYSCALE1 style bit is specified.

**usScale1Spacing** (USHORT)  
Scale spacing.

The spacing between increments, expressed in pixels. It represents the unit that is the smallest division of the scale when the SLS\_PRIMARYSCALE1 style bit is specified. If 0 is specified, the slider automatically calculates the spacing based on the window size and the number of increments specified.

**usScale2Increments** (USHORT)  
Alternate scale increments.

An alternate number of increments to set for the slider control. This number represents the range of values that can be selected within the slider when the SLS\_PRIMARYSCALE2 style bit is specified.

### **usScale2Spacing** (USHORT)

Alternate scale spacing.

An alternate spacing between increments, expressed in pixels. It represents the unit that is the smallest division of the scale when the SLS\_PRIMARYSCALE2 style bit is specified. If 0 is specified, the slider automatically calculates the spacing based on the window size and the number of increments specified.

---

## **SMHSTRUCT**

Send-message-hook structure.

### **Syntax**

```
typedef struct SMHSTRUCT {  
    MPARAM    mp2;  
    MPARAM    mp1;  
    ULONG     msg;  
    HWND      hwnd;  
    ULONG     model;  
} SMHSTRUCT;  
  
typedef SMHSTRUCT *PSMHSTRUCT;
```

### **Fields**

**mp2** (MPARAM)

Parameter 2.

**mp1** (MPARAM)

Parameter 1.

**msg** (ULONG)

Message identity.

**hwnd** (HWND)

Window handle.

**model** (ULONG)

Message identity.

---

## SPBCDATA

Spin Button control data structure.

### Syntax

```
typedef struct _SPBCDATA {
    ULONG      cbSize;
    ULONG      ulTextLimit;
    LONG       lLowerLimit;
    LONG       lUpperLimit;
    ULONG      idMasterSpb;
    PVOID      pHWCtlData;
} SPBCDATA;

typedef SPBCDATA *PSPBCDATA;
```

The SPBCDATA structure is used in WinCreateWindow's *pCtlData* parameter.

When using this structure the SPBM\_SETLIMITS, SPBM\_SETTEXTLIMIT, and SPBM\_SETMASTER messages do not need to be specified.

- *ulTextLimit* and *lLowerLimit* replace SPBM\_SETLIMITS.
- *lUpperLimit* replaces SPBM\_SETTEXTLIMIT.
- *idMasterSpb* replaces SPBM\_SETMASTER.

### Fields

**cbSize** (ULONG)

Size of control block.

**ulTextLimit** (ULONG)

Entryfield text limit.

**lLowerLimit** (LONG)

Spin lower limit (numeric only).

**lUpperLimit** (LONG)

Spin upper limit (numeric only).

**idMasterSpb** (ULONG)

ID of the servant's master spinbutton.

**pHWCtlData** (PVOID)

Reserved for Pen *CtlData*.

---

## **SPLERR**

Error value in the range 0 to 65 535.

### **Syntax**

```
typedef unsigned long SPLERR;
```

---

## **STR16**

String of characters, with an implicit length, in a 16-byte field.

### **Syntax**

```
typedef CHAR STR16[16];
```

---

## **STR32**

String of characters, with an implicit length, in a 32-byte field.

### **Syntax**

```
typedef CHAR STR32[32];
```

---

## **STR64**

String of characters, with an implicit length, in a 64-byte field.

### **Syntax**

```
typedef CHAR STR64[64];
```

---

## STR8

String of 8 characters.

### Syntax

```
typedef CHAR STR8[8];
```

---

## STYLECHANGE

Style-change structure. This structure is returned by the FNTM\_STYLECHANGED message.

All "old" fields describe the style attributes before the user made a change. The other, or "new", parameters describe the style that will be in effect after this is passed to WinDefFontDlgProc. When the "old" and "new" values are the same, the user made no change.

For further details of the parameters, see FONTDLG.

### Syntax

```
typedef struct _STYLECHANGE {
    USHORT    usWeight;
    USHORT    usWeight01d;
    USHORT    usWidth;
    USHORT    usWidth01d;
    ULONG     flType;
    ULONG     flType01d;
    ULONG     flTypeMask;
    ULONG     flTypeMask01d;
    ULONG     flStyle;
    ULONG     flStyle01d;
    ULONG     flStyleMask;
    ULONG     flStyleMask01d;
} STYLECHANGE;

typedef STYLECHANGE *PSTYLECHANGE;
```

## **Fields**

**usWeight** (USHORT)

New weight of font.

**usWeightOld** (USHORT)

Old weight of font.

**usWidth** (USHORT)

New width of font.

**usWidthOld** (USHORT)

Old width of font.

**flType** (ULONG)

New type of font.

**flTypeOld** (ULONG)

Old type of font.

**flTypeMask** (ULONG)

New type mask.

**flTypeMaskOld** (ULONG)

Old type mask.

**flStyle** (ULONG)

New selected style bits.

**flStyleOld** (ULONG)

Old selected style bits.

**flStyleMask** (ULONG)

New mask of style bits to use.

**flStyleMaskOld** (ULONG)

Old mask of style bits to use.

---

## SWBLOCK

Switch-list block structure.

### Syntax

```
typedef struct _SWBLOCK {
    ULONG          cswentry;
    SWENTRY        aswentry[1];
} SWBLOCK;

typedef SWBLOCK *PSWBLOCK;
```

### Fields

**cswentry** (ULONG)

Count of switch list entries.

**aswentry[1]** (SWENTRY)

Switch list entries.

---

## SWCNTRL

Switch-list control block structure.

### Syntax

```
typedef struct _SWCNTRL {
    HWND          hwnd;
    HWND          hwndIcon;
    HPROGRAM      hprog;
    PID           idProcess;
    ULONG         idSession;
    ULONG         uchVisibility;
    ULONG         fbJump;
    CHAR          szSwtitle[MAXNAMEL+4];
    ULONG         bProgType;
} SWCNTRL;

typedef SWCNTRL *PSWCNTRL;
```



## Fields

### **hwnd** (HWND)

Window handle.

### **hwndIcon** (HWND)

Window-handle icon.

### **hprog** (HPROGRAM)

Program handle.

### **idProcess** (PID)

Process identity.

### **idSession** (ULONG)

Session identity.

### **uchVisibility** (ULONG)

Visibility:

|                      |   |
|----------------------|---|
| <b>SWL_VISIBLE</b>   | Visible in startup list   |
| <b>SWL_INVISIBLE</b> | Invisible in startup list   |
| <b>SWL_GRAYED</b>    | Item cannot be switched to (note that it is not actually grayed in the list). |

### **fbJump** (ULONG)

Jump indicator:

|                        |  |
|------------------------|--|
| <b>SWL_JUMPABLE</b>    | Participates in jump sequence          |
| <b>SWL_NOTJUMPABLE</b> | Does not participate in jump sequence. |

### **szSwtitle**[MAXNAMEL+4] (CHAR)

Switch-list control block title (null-terminated).

### **bProgType** (ULONG)

Program type.

Possible values are:

|                           |   |
|---------------------------|---|
| <b>PROG_DEFAULT</b>       | 0 |
| <b>PROG_FULLSCREEN</b>    | 1 |
| <b>PROG_WINDOWABLEVIO</b> | 2 |
| <b>PROG_PM</b>            | 3 |
| <b>PROG_VDM</b>           | 4 |
| <b>PROG_WINDOWEDVDM</b>   | 7 |

Although there are several other program types for WIN-OS/2 programs, these do not show up in this structure. Instead, the **PROG\_VDM** or **PROG\_WINDOWEDVDM** program types are used. For instance, for **PROG\_31\_STDSEAMLESSVDM**, **PROG\_WINDOWEDVDM** is used. This is because all the WIN-OS/2 programs run in DOS sessions. For example, if a program is a windowed WIN-OS/2 program, it runs in a **PROG\_WINDOWEDVDM** session. Likewise, if it's a full-screen WIN-OS/2 program, it runs in a **PROG\_VDM** session.

---

## SWENTRY

Switch-list entry structure.

### Syntax

```
typedef struct _SWENTRY {
    HSWITCH      hswitch;
    SWCNTRL      swctl;
} SWENTRY;

typedef SWENTRY *PSWENTRY;
```

### Fields

#### hswitch (HSWITCH)

Switch-list entry handle.

#### swctl (SWCNTRL)

Switch-list control block structure.

---

## SWP

Set-window-position structure.

### Syntax

```
typedef struct _SWP {
    ULONG      fl;
    LONG       cy;
    LONG       cx;
    LONG       y;
    LONG       x;
    HWND       hwndInsertBehind;
    HWND       hwnd;
    ULONG      u1Reserved1;
    ULONG      u1Reserved2;
} SWP;

typedef SWP *PSWP;
```

## Fields

**fl** (ULONG)

Options.

In alphabetic order:

SWP\_ACTIVATE  
SWP\_DEACTIVATE  
SWP\_HIDE  
SWP\_MAXIMIZE  
SWP\_MINIMIZE  
SWP\_MOVE  
SWP\_NOADJUST  
SWP\_NOERASEWINDOW  
SWP\_NOREDRAW  
SWP\_RESTORE  
SWP\_SHOW  
SWP\_SIZE  
SWP\_ZORDER

**cy** (LONG)

Window height.

**cx** (LONG)

Window width.

**y** (LONG)

Y-coordinate of origin.

**x** (LONG)

X-coordinate of origin.

**hwndInsertBehind** (HWND)

Window behind which this window is placed.

**hwnd** (HWND)

Window handle.

**ulReserved1** (ULONG)

Reserved value, must be 0.

**ulReserved2** (ULONG)

Reserved value, must be 0.

---

## TID

Thread identity.

### Syntax

```
typedef LHANDLE TID;
```

---

## TRACKINFO

Tracking-information structure.

### Syntax

```
typedef struct _TRACKINFO {  
    LONG        cxBorder;  
    LONG        cyBorder;  
    LONG        cxGrid;  
    LONG        cyGrid;  
    LONG        cxKeyboard;  
    LONG        cyKeyboard;  
    RECTL       rc1Track;  
    RECTL       rc1Boundary;  
    POINTL      pt1MinTrackSize;  
    POINTL      pt1MaxTrackSize;  
    ULONG       fs;  
} TRACKINFO;  
  
typedef TRACKINFO *PTRACKINFO;
```

### Fields

#### **cxBorder** (LONG)

Border width.

The width of the left and right tracking sides.

#### **cyBorder** (LONG)

Border height.

The height of the top and bottom tracking sides.

#### **cxGrid** (LONG)

Grid width.

The horizontal bounds of the tracking movements.

**cyGrid** (LONG)

Grid height.

The vertical bounds of the tracking movements.

**cxKeyboard** (LONG)

Character cell width movement for arrow key.

**cyKeyboard** (LONG)

Character cell height movement for arrow key.

**rcTrack** (RECTL)

Starting tracking rectangle.

This is modified as the rectangle is tracked and holds the new tracking position, when tracking is complete.

**rcBoundary** (RECTL)

Boundary rectangle.

This is an absolute bounding rectangle that the tracking rectangle cannot extend; see also TF\_ALLINBOUNDARY.

**ptlMinTrackSize** (POINTL)

Minimum tracking size.

**ptlMaxTrackSize** (POINTL)

Maximum tracking size.

**fs** (ULONG)

Tracking options.

In alphabetic order:

TF\_ALLINBOUNDARY

The default tracking is such that some part of the tracking rectangle is within the bounding rectangle defined by *rcBoundary*. This minimum size is defined by *cxBorder* and *cyBorder*.

If TF\_ALLINBOUNDARY is specified, the tracking is performed so that no part of the tracking rectangle ever falls outside of the bounding rectangle.

TF\_BOTTOM

Track the bottom side of the rectangle.

TF\_GRID

Tracking is restricted to the grid defined by *cxGrid* and *cyGrid*.

TF\_LEFT

Track the left side of the rectangle.

TF\_MOVE

Track all sides of the rectangle.

TF\_RIGHT

Track the right side of the rectangle.

|                  |  |
|------------------|--|
| TF_SETPOINTERPOS | The pointer is repositioned according to other flags as follows:   |
| none             | Pointer is centered in the tracking rectangle.   |
| TF_MOVE          | Pointer is centered in the tracking rectangle.   |
| TF_LEFT          | Pointer is vertically centered at the left of the tracking rectangle.  |
| TF_TOP           | Pointer is horizontally centered at the top of the tracking rectangle.   |
| TF_RIGHT         | Pointer is vertically centered at the right of the tracking rectangle.   |
| TF_BOTTOM        | Pointer is horizontally centered at the bottom of the tracking rectangle.  |
| TF_STANDARD      | <i>cx</i> , <i>cy</i> , <i>cxGrid</i> , and <i>cyGrid</i> are all multiples of <i>cxBorder</i> and <i>cyBorder</i> . |
| TF_TOP           | Track the top side of the rectangle.   |

---

## TREEITEMDESC

Structure that contains icons and bit maps used to represent the state of an expanded or collapsed parent item in the tree name view of a container control.

### Syntax

```
typedef struct _TREEITEMDESC {
    HBITMAP    hbmExpanded;
    HBITMAP    hbmCollapsed;
    HPOINTER   hptrExpanded;
    HPOINTER   hptrCollapsed;
} TREEITEMDESC;

typedef TREEITEMDESC *PTREEITEMDESC;
```

### Fields

#### **hbmExpanded** (HBITMAP)

Expanded bit-map handle.

The handle of the bit map to be used to represent an expanded parent item in the tree name view.

#### **hbmCollapsed** (HBITMAP)

Collapsed bit-map handle.

The handle of the bit map to be used to represent a collapsed parent item in the tree name view.

**hptrExpanded (HPOINTER)**

Expanded icon handle.

The handle of the icon to be used to represent an expanded parent item in the tree name view.

**hptrCollapsed (HPOINTER)**

Collapsed icon handle.

The handle of the icon to be used to represent a collapsed parent item in the tree name view.

---

**TREEMOVE**

Data structure for moving nodes in the tree to a new parent.

**Syntax**

```
typedef struct _TREEMOVE {
    PRECORDCORE    preccMove;
    PRECORDCORE    preccNewParent;
    PRECORDCORE    pRecordOrder;
    BOOL           flMoveSiblings;
} TREEMOVE;

typedef TREEMOVE *PTREEMOVE;
```

**Fields****preccMove (PRECORDCORE)**

Record to be moved.

**preccNewParent (PRECORDCORE)**

New parent for *preccMove*.

### **pRecordOrder** (PRECORDCORE)

Record order for siblings.

Possible values are described in the following list:

**CMA\_FIRST** *preccMove* moves to the FIRST child position of *preccNewParent*. If *preccNewParent* is NULL, *preccMove* becomes the first root level record of the container.

**CMA\_LAST** *preccMove* moves to the LAST child position of *preccNewParent*. If *preccNewParent* is NULL, *preccMove* becomes the last root level record of the container.

**Other** *preccMove* moves after this record in the list of children of *preccNewParent*. If *preccNewParent* is NULL, *preccMove* moves after the record specified by *pRecordOrder* only if that record is also a root level record.

**Note:** This record must currently exist in the list of children of *preccNewParent*.

### **fIMoveSiblings** (BOOL)

Flag indicating whether to move siblings.

**TRUE** All siblings of *preccMove*. that FOLLOW it (from its original location) move to the new parent as well. *pRecordOrder* applies if this flag is TRUE.

**FALSE** Only *preccMove* itself moves to the new parent; all siblings remain with the old parent.

---

## **UCHAR**

Single-byte unsigned character, or unsigned integer in the range 0 through 255.

### **Syntax**

```
typedef unsigned char UCHAR;
```

---

## **ULONG**

Unsigned integer in the range 0 through 4 294 967 295.

### **Syntax**

```
typedef unsigned long ULONG;
```



---

## USERBUTTON

User-button data structure.

### Syntax

```
typedef struct _USERBUTTON {  
    HWND      hwnd;  
    HPS       hps;  
    ULONG     fsState;  
    ULONG     fsStateOld;  
} USERBUTTON;  
  
typedef USERBUTTON *PUSERBUTTON;
```

### Fields

**hwnd** (HWND)

Window handle.

**hps** (HPS)

Presentation-space handle.

**fsState** (ULONG)

New state of user button.

**fsStateOld** (ULONG)

Old state of user button.

---

## USHORT

Unsigned integer in the range 0 through 65 535.

### Syntax

```
typedef unsigned short USHORT;
```

---

## VIOSIZECOUNT

Count of VIO cell sizes. See DevEscape.

### Syntax

```
typedef struct _VIOSIZECOUNT {
    LONG    maxcount;
    LONG    count;
} VIOSIZECOUNT;

typedef VIOSIZECOUNT *PVIOSIZECOUNT;
```

### Fields

**maxcount** (LONG)

Maximum number of VIO cell sizes supported.

**count** (LONG)

Number of VIO cell sizes returned.

---

## VIOFONTCELLSIZE

VIO cell size. See DevEscape.

### Syntax

```
typedef struct _VIOFONTCELLSIZE {
    LONG    cx;
    LONG    cy;
} VIOFONTCELLSIZE;

typedef VIOFONTCELLSIZE *PVIOSIZECOUNT;
```

### Fields

**cx** (LONG)

Cell width.

**cy** (LONG)

Cell height.

---

## VOID

A data area of undefined format.

### Syntax

```
#define VOID void
```

---

## VSCDATA

Structure that contains information about the value set control.

### Syntax

```
typedef struct _VSCDATA {  
    ULONG      cbSize;  
    USHORT     usRowCount;  
    USHORT     usColumnCount;  
} VSCDATA;  
  
typedef VSCDATA *PVSCDATA;
```

### Fields

#### **cbSize** (ULONG)

Data length.

Length of the control data in bytes.

#### **usRowCount** (USHORT)

Number of rows.

The number of rows in the value set control. The minimum number of rows is 1 and the maximum number of rows is 65,535.

#### **usColumnCount** (USHORT)

Number of columns.

The number of columns in the value set control. The minimum number of columns is 1 and the maximum number of columns is 65,535.

---

## VSDRAGINFO

Structure that contains information about direct manipulation actions that occur over the value set control.

### Syntax

```
typedef struct _VSDRAGINFO {
    PDRAGINFO    pDragInfo;
    USHORT       usRow;
    USHORT       usColumn;
} VSDRAGINFO;

typedef VSDRAGINFO *PVSDRAGINFO;
```

### Fields

#### **pDragInfo** (PDRAGINFO)

Pointer to a DRAGINFO structure.

#### **usRow** (USHORT)

Row index.

The index of the row over which the direct manipulation action occurred.

#### **usColumn** (USHORT)

Column index.

The index of the column over which the direct manipulation action occurred.

---

## VSDRAGINIT

Structure that contains information that is used to initialize a direct manipulation action over the value set control.

### Syntax

```
typedef struct _VSDRAGINIT {
    HWND         hwnd;
    LONG         x;
    LONG         y;
    LONG         cx;
    LONG         cy;
    USHORT       usRow;
    USHORT       usColumn;
} VSDRAGINIT;

typedef VSDRAGINIT *PVSDRAGINIT;
```

## Fields

### **hwnd** (HWND)

Value set window handle.

Window handle of the value set control.

### **x** (LONG)

X-coordinate.

X-coordinate of the pointing device pointer in desktop coordinates.

### **y** (LONG)

Y-coordinate.

Y-coordinate of the pointing device pointer in desktop coordinates.

### **cx** (LONG)

X-offset.

X-offset from the hot spot of the pointing device pointer, in pels, to the item origin. The item origin is the lower left corner of the item.

### **cy** (LONG)

Y-offset.

Y-offset from the hot spot of the pointing device pointer, in pels, to the item origin. The item origin is the lower left corner of the item.

### **usRow** (USHORT)

Row index.

The index of the row over which the direct manipulation action occurred.

### **usColumn** (USHORT)

Column index.

The index of the column over which the direct manipulation action occurred.

---

## VSTEXT

Value set text structure. This structure is used with the VM\_QUERYITEM message only. See "VM\_QUERYITEM" on page 26-10 for information about that message.

## Syntax

```
typedef struct _VSTEXT {
    PSZ      pszItemText;
    ULONG    u1BufLen;
} VSTEXT;

typedef VSTEXT *PVSTEXT;
```

## Fields

### **pszItemText** (PSZ)

Pointer to a buffer to copy the string into.

### **ulBufLen** (ULONG)

Buffer size.

Size of the buffer pointed to by the *pszItemText* field.

---

## WNDPARAMS

Window parameters.

## Syntax

```
typedef struct _WNDPARAMS {
    ULONG      fsStatus;
    ULONG      cchText;
    PSZ        pszText;
    ULONG      cbPresParams;
    PVOID      pPresParams;
    ULONG      cbCtlData;
    PVOID      pCtlData;
} WNDPARAMS;

typedef WNDPARAMS *PWNDPARAMS;
```

## Fields

### **fsStatus** (ULONG)

Window parameter selection.

Identifies the window parameters that are to be set or queried:

|                |                            |
|----------------|----------------------------|
| WPM_CBCTLDATA  | Window control data length |
| WPM_CCHTEXT    | Window text length         |
| WPM_CTLDATA    | Window control data        |
| WPM_PRESPARAMS | Presentation parameters    |
| WPM_TEXT       | Window text.               |

### **cchText** (ULONG)

Length of window text.

### **pszText** (PSZ)

Window text.

### **cbPresParams** (ULONG)

Length of presentation parameters.

### **pPresParams** (PVOID)

Presentation parameters.

**cbCtlData** (ULONG)

Length of window class specific data.

**pCtlData** (PVOID)

Window class specific data.

---

## WPOINT

Window-point data structure (long integers). See POINTL for the form of the structure.

### Syntax

```
#define WPOINT POINTL
```

---

## WRECT

Window-rectangle data structure. See RECTL for the form of the structure.

### Syntax

```
#define WRECT RECTL
```

---

## XYWINSIZE

Window position and size structure.

### Syntax

```
typedef struct _XYWINSIZE {  
    SHORT    x;  
    SHORT    y;  
    SHORT    cx;  
    SHORT    cy;  
    USHORT   fsWindow;  
} XYWINSIZE;  
  
typedef XYWINSIZE *PXYWINSIZE;
```

## Fields

**x** (SHORT)

X-coordinate of window origin.

**y** (SHORT)

Y-coordinate of window origin.

**cx** (SHORT)

Window width.

**cy** (SHORT)

Window height.

**fsWindow** (USHORT)

Window options.

The values may be ORed together. For example, an invisible iconic window can be created. Note that if both `XYF_MINIMIZED` and `XYF_MAXIMIZED` are specified, the window is created in a maximized state.

|                              |  |
|------------------------------|--|
| <code>XYF_INVISIBLE</code>   | Create the window initially invisible.   |
| <code>XYF_MAXIMIZED</code>   | Show the window initially maximized.   |
| <code>XYF_MINIMIZED</code>   | Show the window initially iconic.  |
| <code>XYF_NOAUTOCLOSE</code> | Do not close the window automatically when the VIO application terminates. This parameter is ignored unless the program is a VIO-windowed application. |
| <code>XYF_NORMAL</code>      | Create the window visible, with a size and position as specified. This is the default.   |





---

## Appendix B. Error Codes

This section lists PM errors returned by WinGetLastError in order of their error numbers. For explanations of these errors, see Appendix C, "Error Explanations" on page C-1.

| <b>Error Number</b> | <b>Error Constant</b>           |
|---------------------|---------------------------------|
| 0x0000              | PMERR_OK                        |
| 0x0836              | NERR_NetNotStarted              |
| 0x0845              | NERR_RedirectedPath             |
| 0x084B              | NERR_BufTooSmall                |
| 0x085E              | NERR_InvalidAPI                 |
| 0x0866              | NERR_QNotFound                  |
| 0x0867              | NERR_JobNotFound                |
| 0x0868              | NERR_DestNotFound               |
| 0x0869              | NERR_DestExists                 |
| 0x086A              | NERR_QExists                    |
| 0x086B              | NERR_QNoRoom                    |
| 0x086C              | NERR_JobNoRoom                  |
| 0x086D              | NERR_DestNoRoom                 |
| 0x086E              | NERR_DestIdle                   |
| 0x086F              | NERR_DestInvalidOp              |
| 0x0871              | NERR_SpoolerNotLoaded           |
| 0x0872              | NERR_DestInvalidState           |
| 0x0874              | NERR_JobInvalidState            |
| 0x0875              | NERR_SpoolNoMemory              |
| 0x0876              | NERR_DriverNotFound             |
| 0x0877              | NERR_DataTypeInvalid            |
| 0x0878              | NERR_ProcNotFound               |
| 0x0925              | NERR_BadDev                     |
| 0x0927              | NERR_CommDevInUse               |
| 0x092F              | NERR_InvalidComputer            |
| 0x0961              | NERR_OpenFiles                  |
| 0x0965              | NERR_LocalDrive                 |
| 0x1001              | PMERR_INVALID_HWND              |
| 0x1001              | HMERR_NO_FRAME_WND_IN_CHAIN     |
| 0x1002              | PMERR_INVALID_HMQ               |
| 0x1002              | HMERR_INVALID_ASSOC_APP_WND     |
| 0x1003              | PMERR_PARAMETER_OUT_OF_RANGE    |
| 0x1003              | HMERR_INVALID_ASSOC_HELP_INST   |
| 0x1004              | PMERR_WINDOW_LOCK_UNDERFLOW     |
| 0x1004              | HMERR_INVALID_DESTROY_HELP_INST |
| 0x1005              | PMERR_WINDOW_LOCK_OVERFLOW      |
| 0x1005              | HMERR_NO_HELP_INST_IN_CHAIN     |
| 0x1006              | PMERR_BAD_WINDOW_LOCK_COUNT     |
| 0x1006              | HMERR_INVALID_HELP_INSTANCE_HDL |
| 0x1007              | PMERR_WINDOW_NOT_LOCKED         |
| 0x1007              | HMERR_INVALID_QUERY_APP_WND     |
| 0x1008              | PMERR_INVALID_SELECTOR          |

|        |                                |
|--------|--------------------------------|
| 0x1008 | HMERR_HELP_INST_CALLED_INVALID |
| 0x1009 | PMERR_CALL_FROM_WRONG_THREAD   |
| 0x1009 | HMERR_HELPTABLE_UNDEFINE       |
| 0x100A | PMERR_RESOURCE_NOT_FOUND       |
| 0x100A | HMERR_HELP_INSTANCE_UNDEFINE   |
| 0x100B | PMERR_INVALID_STRING_PARM      |
| 0x100B | HMERR_HELPITEM_NOT_FOUND       |
| 0x100C | PMERR_INVALID_HHEAP            |
| 0x100C | HMERR_INVALID_HELPSUBITEM_SIZE |
| 0x100D | PMERR_INVALID_HEAP_POINTER     |
| 0x100D | HMERR_HELPSUBITEM_NOT_FOUND    |
| 0x100E | PMERR_INVALID_HEAP_SIZE_PARM   |
| 0x100F | PMERR_INVALID_HEAP_SIZE        |
| 0x1010 | PMERR_INVALID_HEAP_SIZE_WORD   |
| 0x1011 | PMERR_HEAP_OUT_OF_MEMORY       |
| 0x1012 | PMERR_HEAP_MAX_SIZE_REACHED    |
| 0x1013 | PMERR_INVALID_HATOMTBL         |
| 0x1014 | PMERR_INVALID_ATOM             |
| 0x1015 | PMERR_INVALID_ATOM_NAME        |
| 0x1016 | PMERR_INVALID_INTEGER_ATOM     |
| 0x1017 | PMERR_ATOM_NAME_NOT_FOUND      |
| 0x1018 | PMERR_QUEUE_TOO_LARGE          |
| 0x1019 | PMERR_INVALID_FLAG             |
| 0x101A | PMERR_INVALID_HACCEL           |
| 0x101B | PMERR_INVALID_HPTR             |
| 0x101C | PMERR_INVALID_HENUM            |
| 0x101D | PMERR_INVALID_SRC_CODEPAGE     |
| 0x101E | PMERR_INVALID_DST_CODEPAGE     |
| 0x101F | PMERR_UNKNOWN_COMPONENT_ID     |
| 0x1020 | PMERR_UNKNOWN_ERROR_CODE       |
| 0x1021 | PMERR_SEVERITY_LEVELS          |
| 0x1034 | PMERR_INVALID_RESOURCE_FORMAT  |
| 0x1035 | WINDBG_WINDOW_UNLOCK_WAIT      |
| 0x1036 | PMERR_NO_MSG_QUEUE             |
| 0x1037 | PMERR_WIN_DEBUGMSG             |
| 0x1038 | PMERR_QUEUE_FULL               |
| 0x1039 | PMERR_LIBRARY_LOAD_FAILED      |
| 0x103A | PMERR_PROCEDURE_LOAD_FAILED    |
| 0x103B | PMERR_LIBRARY_DELETE_FAILED    |
| 0x103C | PMERR_PROCEDURE_DELETE_FAILED  |
| 0x103D | PMERR_ARRAY_TOO_LARGE          |
| 0x103E | PMERR_ARRAY_TOO_SMALL          |
| 0x103F | PMERR_DATATYPE_ENTRY_BAD_INDEX |
| 0x1040 | PMERR_DATATYPE_ENTRY_CTL_BAD   |
| 0x1041 | PMERR_DATATYPE_ENTRY_CTL_MISS  |
| 0x1042 | PMERR_DATATYPE_ENTRY_INVALID   |
| 0x1043 | PMERR_DATATYPE_ENTRY_NOT_NUM   |
| 0x1044 | PMERR_DATATYPE_ENTRY_NOT_OFF   |

|        |                                |
|--------|--------------------------------|
| 0x1045 | PMERR_DATATYPE_INVALID         |
| 0x1046 | PMERR_DATATYPE_NOT_UNIQUE      |
| 0x1047 | PMERR_DATATYPE_TOO_LONG        |
| 0x1048 | PMERR_DATATYPE_TOO_SMALL       |
| 0x1049 | PMERR_DIRECTION_INVALID        |
| 0x104A | PMERR_INVALID_HAB              |
| 0x104D | PMERR_INVALID_HSTRUCT          |
| 0x104E | PMERR_LENGTH_TOO_SMALL         |
| 0x104F | PMERR_MSGID_TOO_SMALL          |
| 0x1050 | PMERR_NO_HANDLE_ALLOC          |
| 0x1051 | PMERR_NOT_IN_A_PM_SESSION      |
| 0x1052 | PMERR_MSG_QUEUE_ALREADY_EXISTS |
| 0x1055 | PMERR_OLD_RESOURCE             |
| 0x1056 | PMERR_WPDSEVER_IS_ACTIVE       |
| 0x1057 | PMERR_WPDSEVER_NOT_STARTED     |
| 0x1058 | PMERR_SOMDD_IS_ACTIVE          |
| 0x1059 | PMERR_SOMDD_NOT_STARTED        |
| 0x1101 | PMERR_INVALID_PIB              |
| 0x1102 | PMERR_INSUFF_SPACE_TO_ADD      |
| 0x1103 | PMERR_INVALID_GROUP_HANDLE     |
| 0x1104 | PMERR_DUPLICATE_TITLE          |
| 0x1105 | PMERR_INVALID_TITLE            |
| 0x1106 | PMERR_INVALID_TARGET_HANDLE    |
| 0x1107 | PMERR_HANDLE_NOT_IN_GROUP      |
| 0x1108 | PMERR_INVALID_PATH_STATEMENT   |
| 0x1109 | PMERR_NO_PROGRAM_FOUND         |
| 0x110A | PMERR_INVALID_BUFFER_SIZE      |
| 0x110B | PMERR_BUFFER_TOO_SMALL         |
| 0x110C | PMERR_PL_INITIALISATION_FAIL   |
| 0x110D | PMERR_CANT_DESTROY_SYS_GROUP   |
| 0x110E | PMERR_INVALID_TYPE_CHANGE      |
| 0x110F | PMERR_INVALID_PROGRAM_HANDLE   |
| 0x1110 | PMERR_NOT_CURRENT_PL_VERSION   |
| 0x1111 | PMERR_INVALID_CIRCULAR_REF     |
| 0x1112 | PMERR_MEMORY_ALLOCATION_ERR    |
| 0x1113 | PMERR_MEMORY_DEALLOCATION_ERR  |
| 0x1114 | PMERR_TASK_HEADER_TOO_BIG      |
| 0x1115 | PMERR_INVALID_INI_FILE_HANDLE  |
| 0x1116 | PMERR_MEMORY_SHARE             |
| 0x1117 | PMERR_OPEN_QUEUE               |
| 0x1118 | PMERR_CREATE_QUEUE             |
| 0x1119 | PMERR_WRITE_QUEUE              |
| 0x111A | PMERR_READ_QUEUE               |
| 0x111B | PMERR_CALL_NOT_EXECUTED        |
| 0x111C | PMERR_UNKNOWN_APIPKT           |
| 0x111D | PMERR_INITHREAD_EXISTS         |
| 0x111E | PMERR_CREATE_THREAD            |
| 0x111F | PMERR_NO_HK_PROFILE_INSTALLED  |

|        |                                 |
|--------|---------------------------------|
| 0x1120 | PMERR_INVALID_DIRECTORY         |
| 0x1121 | PMERR_WILDCARD_IN_FILENAME      |
| 0x1122 | PMERR_FILENAME_BUFFER_FULL      |
| 0x1123 | PMERR_FILENAME_TOO_LONG         |
| 0x1124 | PMERR_INI_FILE_IS_SYS_OR_USER   |
| 0x1125 | PMERR_BROADCAST_PLMSG           |
| 0x1126 | PMERR_190_INIT_DONE             |
| 0x1127 | PMERR_HMOD_FOR_PMSHAPI          |
| 0x1128 | PMERR_SET_HK_PROFILE            |
| 0x1129 | PMERR_API_NOT_ALLOWED           |
| 0x112A | PMERR_INI_STILL_OPEN            |
| 0x112B | PMERR_PROGDETAILS_NOT_IN_INI    |
| 0x112C | PMERR_PIBSTRUCT_NOT_IN_INI      |
| 0x112D | PMERR_INVALID_DISKPROGDETAILS   |
| 0x112E | PMERR_PROGDETAILS_READ_FAILURE  |
| 0x112F | PMERR_PROGDETAILS_WRITE_FAILURE |
| 0x1130 | PMERR_PROGDETAILS_QSIZE_FAILURE |
| 0x1131 | PMERR_INVALID_PROGDETAILS       |
| 0x1132 | PMERR_SHEPROFILEHOOK_NOT_FOUND  |
| 0x1133 | PMERR_190PLCONVERTED            |
| 0x1134 | PMERR_FAILED_TO_CONVERT_INI_PL  |
| 0x1135 | PMERR_PMSHAPI_NOT_INITIALISED   |
| 0x1136 | PMERR_INVALID_SHELL_API_HOOK_ID |
| 0x1200 | PMERR_DOS_ERROR                 |
| 0x1201 | PMERR_NO_SPACE                  |
| 0x1202 | PMERR_INVALID_SWITCH_HANDLE     |
| 0x1203 | PMERR_NO_HANDLE                 |
| 0x1204 | PMERR_INVALID_PROCESS_ID        |
| 0x1205 | PMERR_NOT_SHELL                 |
| 0x1206 | PMERR_INVALID_WINDOW            |
| 0x1207 | PMERR_INVALID_POST_MSG          |
| 0x1208 | PMERR_INVALID_PARAMETERS        |
| 0x1208 | PMERR_INVALID_PARAMETERS        |
| 0x1209 | PMERR_INVALID_PROGRAM_TYPE      |
| 0x120A | PMERR_NOT_EXTENDED_FOCUS        |
| 0x120B | PMERR_INVALID_SESSION_ID        |
| 0x120C | PMERR_SMG_INVALID_ICON_FILE     |
| 0x120D | PMERR_SMG_ICON_NOT_CREATED      |
| 0x120E | PMERR_SHL_DEBUG                 |
| 0x1301 | PMERR_OPENING_INI_FILE          |
| 0x1302 | PMERR_INI_FILE_CORRUPT          |
| 0x1303 | PMERR_INVALID_PARM              |
| 0x1304 | PMERR_NOT_IN_IDX                |
| 0x1305 | PMERR_NO_ENTRIES_IN_GROUP       |
| 0x1306 | PMERR_INI_WRITE_FAIL            |
| 0x1307 | PMERR_IDX_FULL                  |
| 0x1308 | PMERR_INI_PROTECTED             |
| 0x1309 | PMERR_MEMORY_ALLOC              |

|        |                                 |
|--------|---------------------------------|
| 0x130A | PMERR_INI_INIT_ALREADY_DONE     |
| 0x130B | PMERR_INVALID_INTEGER           |
| 0x130C | PMERR_INVALID_ASCII             |
| 0x130D | PMERR_CAN_NOT_CALL_SPOOLER      |
| 0x130D | PMERR_VALIDATION_REJECTED       |
| 0x1401 | PMERR_WARNING_WINDOW_NOT_KILLED |
| 0x1402 | PMERR_ERROR_INVALID_WINDOW      |
| 0x1403 | PMERR_ALREADY_INITIALIZED       |
| 0x1405 | PMERR_MSG_PROG_NO_MOU           |
| 0x1406 | PMERR_MSG_PROG_NON_RECOV        |
| 0x1407 | PMERR_WINCONV_INVALID_PATH      |
| 0x1408 | PMERR_PI_NOT_INITIALISED        |
| 0x1409 | PMERR_PL_NOT_INITIALISED        |
| 0x140A | PMERR_NO_TASK_MANAGER           |
| 0x140B | PMERR_SAVE_NOT_IN_PROGRESS      |
| 0x140C | PMERR_NO_STACK_SPACE            |
| 0x140D | PMERR_INVALID_COLR_FIELD        |
| 0x140E | PMERR_INVALID_COLR_VALUE        |
| 0x140F | PMERR_COLR_WRITE                |
| 0x1501 | PMERR_TARGET_FILE_EXISTS        |
| 0x1502 | PMERR_SOURCE_SAME_AS_TARGET     |
| 0x1503 | PMERR_SOURCE_FILE_NOT_FOUND     |
| 0x1504 | PMERR_INVALID_NEW_PATH          |
| 0x1505 | PMERR_TARGET_FILE_NOT_FOUND     |
| 0x1506 | PMERR_INVALID_DRIVE_NUMBER      |
| 0x1507 | PMERR_NAME_TOO_LONG             |
| 0x1508 | PMERR_NOT_ENOUGH_ROOM_ON_DISK   |
| 0x1509 | PMERR_NOT_ENOUGH_MEM            |
| 0x150B | PMERR_LOG_DRV_DOES_NOT_EXIST    |
| 0x150C | PMERR_INVALID_DRIVE             |
| 0x150D | PMERR_ACCESS_DENIED             |
| 0x150E | PMERR_NO_FIRST_SLASH            |
| 0x150F | PMERR_READ_ONLY_FILE            |
| 0x151F | PMERR_GROUP_PROTECTED           |
| 0x152F | PMERR_INVALID_PROGRAM_CATEGORY  |
| 0x1530 | PMERR_INVALID_APPL              |
| 0x1531 | PMERR_CANNOT_START              |
| 0x1532 | PMERR_STARTED_IN_BACKGROUND     |
| 0x1533 | PMERR_INVALID_HAPP              |
| 0x1534 | PMERR_CANNOT_STOP               |
| 0x1601 | PMERR_INTERNAL_ERROR_1          |
| 0x1602 | PMERR_INTERNAL_ERROR_2          |
| 0x1603 | PMERR_INTERNAL_ERROR_3          |
| 0x1604 | PMERR_INTERNAL_ERROR_4          |
| 0x1605 | PMERR_INTERNAL_ERROR_5          |
| 0x1606 | PMERR_INTERNAL_ERROR_6          |
| 0x1607 | PMERR_INTERNAL_ERROR_7          |
| 0x1608 | PMERR_INTERNAL_ERROR_8          |

|        |                                |
|--------|--------------------------------|
| 0x1609 | PMERR_INTERNAL_ERROR_9         |
| 0x160A | PMERR_INTERNAL_ERROR_10        |
| 0x160B | PMERR_INTERNAL_ERROR_11        |
| 0x160C | PMERR_INTERNAL_ERROR_12        |
| 0x160D | PMERR_INTERNAL_ERROR_13        |
| 0x160E | PMERR_INTERNAL_ERROR_14        |
| 0x160F | PMERR_INTERNAL_ERROR_15        |
| 0x1610 | PMERR_INTERNAL_ERROR_16        |
| 0x1611 | PMERR_INTERNAL_ERROR_17        |
| 0x1612 | PMERR_INTERNAL_ERROR_18        |
| 0x1613 | PMERR_INTERNAL_ERROR_19        |
| 0x1614 | PMERR_INTERNAL_ERROR_20        |
| 0x1615 | PMERR_INTERNAL_ERROR_21        |
| 0x1616 | PMERR_INTERNAL_ERROR_22        |
| 0x1617 | PMERR_INTERNAL_ERROR_23        |
| 0x1618 | PMERR_INTERNAL_ERROR_24        |
| 0x1619 | PMERR_INTERNAL_ERROR_25        |
| 0x161A | PMERR_INTERNAL_ERROR_26        |
| 0x161B | PMERR_INTERNAL_ERROR_27        |
| 0x161C | PMERR_INTERNAL_ERROR_28        |
| 0x161D | PMERR_INTERNAL_ERROR_29        |
| 0x1630 | PMERR_INVALID_FREE_MESSAGE_ID  |
| 0x1641 | PMERR_FUNCTION_NOT_SUPPORTED   |
| 0x1642 | PMERR_INVALID_ARRAY_COUNT      |
| 0x1643 | PMERR_INVALID_LENGTH           |
| 0x1644 | PMERR_INVALID_BUNDLE_TYPE      |
| 0x1645 | PMERR_INVALID_PARAMETER        |
| 0x1646 | PMERR_INVALID_NUMBER_OF_PARMS  |
| 0x1647 | PMERR_GREATER_THAN_64K         |
| 0x1648 | PMERR_INVALID_PARAMETER_TYPE   |
| 0x1649 | PMERR_NEGATIVE_STRCOND_DIM     |
| 0x164A | PMERR_INVALID_NUMBER_OF_TYPES  |
| 0x164B | PMERR_INCORRECT_HSTRUCT        |
| 0x164C | PMERR_INVALID_ARRAY_SIZE       |
| 0x164D | PMERR_INVALID_CONTROL_DATATYPE |
| 0x164E | PMERR_INCOMPLETE_CONTROL_SEQU  |
| 0x164F | PMERR_INVALID_DATATYPE         |
| 0x1650 | PMERR_INCORRECT_DATATYPE       |
| 0x1651 | PMERR_NOT_SELF_DESCRIBING_DTYP |
| 0x1652 | PMERR_INVALID_CTRL_SEQ_INDEX   |
| 0x1653 | PMERR_INVALID_TYPE_FOR_LENGTH  |
| 0x1654 | PMERR_INVALID_TYPE_FOR_OFFSET  |
| 0x1655 | PMERR_INVALID_TYPE_FOR_MPARAM  |
| 0x1656 | PMERR_INVALID_MESSAGE_ID       |
| 0x1657 | PMERR_C_LENGTH_TOO_SMALL       |
| 0x1658 | PMERR_APPL_STRUCTURE_TOO_SMALL |
| 0x1659 | PMERR_INVALID_ERRORINFO_HANDLE |
| 0x165A | PMERR_INVALID_CHARACTER_INDEX  |

|        |                              |
|--------|------------------------------|
| 0x1700 | WPERR_PROTECTED_CLASS        |
| 0x1701 | WPERR_INVALID_CLASS          |
| 0x1702 | WPERR_INVALID_SUPERCLASS     |
| 0x1703 | WPERR_NO_MEMORY              |
| 0x1704 | WPERR_SEMAPHORE_ERROR        |
| 0x1705 | WPERR_BUFFER_TOO_SMALL       |
| 0x1706 | WPERR_CLSLOADMOD_FAILED      |
| 0x1707 | WPERR_CLSPROCADDR_FAILED     |
| 0x1708 | WPERR_OBJWORD_LOCATION       |
| 0x1709 | WPERR_INVALID_OBJECT         |
| 0x170A | WPERR_MEMORY_CLEANUP         |
| 0x170B | WPERR_INVALID_MODULE         |
| 0x170C | WPERR_INVALID_OLDCLASS       |
| 0x170D | WPERR_INVALID_NEWCLASS       |
| 0x170E | WPERR_NOT_IMMEDIATE_CHILD    |
| 0x170F | WPERR_NOT_WORKPLACE_CLASS    |
| 0x1710 | WPERR_CANT_REPLACE_METACLS   |
| 0x1711 | WPERR_INI_FILE_WRITE         |
| 0x1712 | WPERR_INVALID_FOLDER         |
| 0x1713 | WPERR_BUFFER_OVERFLOW        |
| 0x1714 | WPERR_OBJECT_NOT_FOUND       |
| 0x1715 | WPERR_INVALID_HFIND          |
| 0x1716 | WPERR_INVALID_COUNT          |
| 0x1717 | WPERR_INVALID_BUFFER         |
| 0x1718 | WPERR_ALREADY_EXISTS         |
| 0x1719 | WPERR_INVALID_FLAGS          |
| 0x1720 | WPERR_INVALID_OBJECTID       |
| 0x1721 | WPERR_INVALID_TARGET_OBJECT  |
| 0x1F00 | PMERR_NOT_DRAGGING           |
| 0x2001 | PMERR_ALREADY_IN_AREA        |
| 0x2001 | HMERR_INDEX_NOT_FOUND        |
| 0x2002 | PMERR_ALREADY_IN_ELEMENT     |
| 0x2002 | HMERR_CONTENT_NOT_FOUND      |
| 0x2003 | PMERR_ALREADY_IN_PATH        |
| 0x2003 | HMERR_OPEN_LIB_FILE          |
| 0x2004 | PMERR_ALREADY_IN_SEG         |
| 0x2004 | HMERR_READ_LIB_FILE          |
| 0x2005 | PMERR_AREA_INCOMPLETE        |
| 0x2005 | HMERR_CLOSE_LIB_FILE         |
| 0x2006 | PMERR_BASE_ERROR             |
| 0x2006 | HMERR_INVALID_LIB_FILE       |
| 0x2007 | PMERR_BITBLT_LENGTH_EXCEEDED |
| 0x2007 | HMERR_NO_MEMORY              |
| 0x2008 | PMERR_BITMAP_IN_USE          |
| 0x2008 | HMERR_ALLOCATE_SEGMENT       |
| 0x2009 | PMERR_BITMAP_IS_SELECTED     |
| 0x2009 | HMERR_FREE_MEMORY            |
| 0x200A | PMERR_BITMAP_NOT_FOUND       |



|        |                                |
|--------|--------------------------------|
| 0x200B | PMERR_BITMAP_NOT_SELECTED      |
| 0x200C | PMERR_BOUNDS_OVERFLOW          |
| 0x200D | PMERR_CALLED_SEG_IS_CHAINED    |
| 0x200E | PMERR_CALLED_SEG_IS_CURRENT    |
| 0x200F | PMERR_CALLED_SEG_NOT_FOUND     |
| 0x2010 | PMERR_CANNOT_DELETE_ALL_DATA   |
| 0x2010 | HMERR_PANEL_NOT_FOUND          |
| 0x2011 | PMERR_CANNOT_REPLACE_ELEMENT_0 |
| 0x2011 | HMERR_DATABASE_NOT_OPEN        |
| 0x2012 | PMERR_COL_TABLE_NOT_REALIZABLE |
| 0x2013 | PMERR_COL_TABLE_NOT_REALIZED   |
| 0x2013 | HMERR_LOAD_DLL                 |
| 0x2014 | PMERR_COORDINATE_OVERFLOW      |
| 0x2015 | PMERR_CORR_FORMAT_MISMATCH     |
| 0x2016 | PMERR_DATA_TOO_LONG            |
| 0x2017 | PMERR_DC_IS_ASSOCIATED         |
| 0x2018 | PMERR_DESC_STRING_TRUNCATED    |
| 0x2019 | PMERR_DEVICE_DRIVER_ERROR_1    |
| 0x201A | PMERR_DEVICE_DRIVER_ERROR_2    |
| 0x201B | PMERR_DEVICE_DRIVER_ERROR_3    |
| 0x201C | PMERR_DEVICE_DRIVER_ERROR_4    |
| 0x201D | PMERR_DEVICE_DRIVER_ERROR_5    |
| 0x201E | PMERR_DEVICE_DRIVER_ERROR_6    |
| 0x201F | PMERR_DEVICE_DRIVER_ERROR_7    |
| 0x2020 | PMERR_DEVICE_DRIVER_ERROR_8    |
| 0x2021 | PMERR_DEVICE_DRIVER_ERROR_9    |
| 0x2022 | PMERR_DEVICE_DRIVER_ERROR_10   |
| 0x2023 | PMERR_DEV_FUNC_NOT_INSTALLED   |
| 0x2024 | PMERR_DOSOPEN_FAILURE          |
| 0x2025 | PMERR_DOSREAD_FAILURE          |
| 0x2026 | PMERR_DRIVER_NOT_FOUND         |
| 0x2027 | PMERR_DUP_SEG                  |
| 0x2028 | PMERR_DYNAMIC_SEG_SEQ_ERROR    |
| 0x2029 | PMERR_DYNAMIC_SEG_ZERO_INV     |
| 0x202A | PMERR_ELEMENT_INCOMPLETE       |
| 0x202B | PMERR_ESC_CODE_NOT_SUPPORTED   |
| 0x202C | PMERR_EXCEEDS_MAX_SEG_LENGTH   |
| 0x202D | PMERR_FONT_AND_MODE_MISMATCH   |
| 0x202E | PMERR_FONT_FILE_NOT_LOADED     |
| 0x202F | PMERR_FONT_NOT_LOADED          |
| 0x2030 | PMERR_FONT_TOO_BIG             |
| 0x2031 | PMERR_HARDWARE_INIT_FAILURE    |
| 0x2032 | PMERR_HBITMAP_BUSY             |
| 0x2033 | PMERR_HDC_BUSY                 |
| 0x2034 | PMERR_HRGN_BUSY                |
| 0x2035 | PMERR_HUGE_FONTS_NOT_SUPPORTED |
| 0x2036 | PMERR_ID_HAS_NO_BITMAP         |
| 0x2037 | PMERR_IMAGE_INCOMPLETE         |

|        |                               |
|--------|-------------------------------|
| 0x2038 | PMERR_INCOMPAT_COLOR_FORMAT   |
| 0x2039 | PMERR_INCOMPAT_COLOR_OPTIONS  |
| 0x203A | PMERR_INCOMPATIBLE_BITMAP     |
| 0x203B | PMERR_INCOMPATIBLE_METAFILE   |
| 0x203C | PMERR_INCORRECT_DC_TYPE       |
| 0x203D | PMERR_INSUFFICIENT_DISK_SPACE |
| 0x203E | PMERR_INSUFFICIENT_MEMORY     |
| 0x203F | PMERR_INV_ANGLE_PARM          |
| 0x2040 | PMERR_INV_ARC_CONTROL         |
| 0x2041 | PMERR_INV_AREA_CONTROL        |
| 0x2042 | PMERR_INV_ARC_POINTS          |
| 0x2043 | PMERR_INV_ATTR_MODE           |
| 0x2044 | PMERR_INV_BACKGROUND_COL_ATTR |
| 0x2045 | PMERR_INV_BACKGROUND_MIX_ATTR |
| 0x2046 | PMERR_INV_BITBLT_MIX          |
| 0x2047 | PMERR_INV_BITBLT_STYLE        |
| 0x2048 | PMERR_INV_BITMAP_DIMENSION    |
| 0x2049 | PMERR_INV_BOX_CONTROL         |
| 0x204A | PMERR_INV_BOX_ROUNDING_PARM   |
| 0x204B | PMERR_INV_CHAR_ANGLE_ATTR     |
| 0x204C | PMERR_INV_CHAR_DIRECTION_ATTR |
| 0x204D | PMERR_INV_CHAR_MODE_ATTR      |
| 0x204E | PMERR_INV_CHAR_POS_OPTIONS    |
| 0x204F | PMERR_INV_CHAR_SET_ATTR       |
| 0x2050 | PMERR_INV_CHAR_SHEAR_ATTR     |
| 0x2051 | PMERR_INV_CLIP_PATH_OPTIONS   |
| 0x2052 | PMERR_INV_CODEPAGE            |
| 0x2053 | PMERR_INV_COLOR_ATTR          |
| 0x2054 | PMERR_INV_COLOR_DATA          |
| 0x2055 | PMERR_INV_COLOR_FORMAT        |
| 0x2056 | PMERR_INV_COLOR_INDEX         |
| 0x2057 | PMERR_INV_COLOR_OPTIONS       |
| 0x2058 | PMERR_INV_COLOR_START_INDEX   |
| 0x2059 | PMERR_INV_COORD_OFFSET        |
| 0x205A | PMERR_INV_COORD_SPACE         |
| 0x205B | PMERR_INV_COORDINATE          |
| 0x205C | PMERR_INV_CORRELATE_DEPTH     |
| 0x205D | PMERR_INV_CORRELATE_TYPE      |
| 0x205E | PMERR_INV_CURSOR_BITMAP       |
| 0x205F | PMERR_INV_DC_DATA             |
| 0x2060 | PMERR_INV_DC_TYPE             |
| 0x2061 | PMERR_INV_DEVICE_NAME         |
| 0x2062 | PMERR_INV_DEV_MODES_OPTIONS   |
| 0x2063 | PMERR_INV_DRAW_CONTROL        |
| 0x2064 | PMERR_INV_DRAW_VALUE          |
| 0x2065 | PMERR_INV_DRAWING_MODE        |
| 0x2066 | PMERR_INV_DRIVER_DATA         |
| 0x2067 | PMERR_INV_DRIVER_NAME         |

|        |                                |
|--------|--------------------------------|
| 0x2068 | PMERR_INV_DRAW_BORDER_OPTION   |
| 0x2069 | PMERR_INV_EDIT_MODE            |
| 0x206A | PMERR_INV_ELEMENT_OFFSET       |
| 0x206B | PMERR_INV_ELEMENT_POINTER      |
| 0x206C | PMERR_INV_END_PATH_OPTIONS     |
| 0x206D | PMERR_INV_ESC_CODE             |
| 0x206E | PMERR_INV_ESCAPE_DATA          |
| 0x206F | PMERR_INV_EXTENDED_LCID        |
| 0x2070 | PMERR_INV_FILL_PATH_OPTIONS    |
| 0x2071 | PMERR_INV_FIRST_CHAR           |
| 0x2072 | PMERR_INV_FONT_ATTRS           |
| 0x2073 | PMERR_INV_FONT_FILE_DATA       |
| 0x2074 | PMERR_INV_FOR_THIS_DC_TYPE     |
| 0x2075 | PMERR_INV_FORMAT_CONTROL       |
| 0x2076 | PMERR_INV_FORMS_CODE           |
| 0x2077 | PMERR_INV_FONTDEF              |
| 0x2078 | PMERR_INV_GEOM_LINE_WIDTH_ATTR |
| 0x2079 | PMERR_INV_GETDATA_CONTROL      |
| 0x207A | PMERR_INV_GRAPHICS_FIELD       |
| 0x207B | PMERR_INV_HBITMAP              |
| 0x207C | PMERR_INV_HDC                  |
| 0x207D | PMERR_INV_HJOURNAL             |
| 0x207E | PMERR_INV_HMF                  |
| 0x207F | PMERR_INV_HPS                  |
| 0x2080 | PMERR_INV_HRGN                 |
| 0x2081 | PMERR_INV_ID                   |
| 0x2082 | PMERR_INV_IMAGE_DATA_LENGTH    |
| 0x2083 | PMERR_INV_IMAGE_DIMENSION      |
| 0x2084 | PMERR_INV_IMAGE_FORMAT         |
| 0x2085 | PMERR_INV_IN_AREA              |
| 0x2086 | PMERR_INV_IN_CALLED_SEG        |
| 0x2087 | PMERR_INV_IN_CURRENT_EDIT_MODE |
| 0x2088 | PMERR_INV_IN_DRAW_MODE         |
| 0x2089 | PMERR_INV_IN_ELEMENT           |
| 0x208A | PMERR_INV_IN_IMAGE             |
| 0x208B | PMERR_INV_IN_PATH              |
| 0x208C | PMERR_INV_IN_RETAIN_MODE       |
| 0x208D | PMERR_INV_IN_SEG               |
| 0x208E | PMERR_INV_IN_VECTOR_SYMBOL     |
| 0x208F | PMERR_INV_INFO_TABLE           |
| 0x2090 | PMERR_INV_JOURNAL_OPTION       |
| 0x2091 | PMERR_INV_KERNING_FLAGS        |
| 0x2092 | PMERR_INV_LENGTH_OR_COUNT      |
| 0x2093 | PMERR_INV_LINE_END_ATTR        |
| 0x2094 | PMERR_INV_LINE_JOIN_ATTR       |
| 0x2095 | PMERR_INV_LINE_TYPE_ATTR       |
| 0x2096 | PMERR_INV_LINE_WIDTH_ATTR      |
| 0x2097 | PMERR_INV_LOGICAL_ADDRESS      |

|        |                                |
|--------|--------------------------------|
| 0x2098 | PMERR_INV_MARKER_BOX_ATTR      |
| 0x2099 | PMERR_INV_MARKER_SET_ATTR      |
| 0x209A | PMERR_INV_MARKER_SYMBOL_ATTR   |
| 0x209B | PMERR_INV_MATRIX_ELEMENT       |
| 0x209C | PMERR_INV_MAX_HITS             |
| 0x209D | PMERR_INV_METAFILE             |
| 0x209E | PMERR_INV_METAFILE_LENGTH      |
| 0x209F | PMERR_INV_METAFILE_OFFSET      |
| 0x20A0 | PMERR_INV_MICROPS_DRAW_CONTROL |
| 0x20A1 | PMERR_INV_MICROPS_FUNCTION     |
| 0x20A2 | PMERR_INV_MICROPS_ORDER        |
| 0x20A3 | PMERR_INV_MIX_ATTR             |
| 0x20A4 | PMERR_INV_MODE_FOR_OPEN_DYN    |
| 0x20A5 | PMERR_INV_MODE_FOR_REOPEN_SEG  |
| 0x20A6 | PMERR_INV_MODIFY_PATH_MODE     |
| 0x20A7 | PMERR_INV_MULTIPLIER           |
| 0x20A8 | PMERR_INV_NESTED_FIGURES       |
| 0x20A9 | PMERR_INV_OR_INCOMPAT_OPTIONS  |
| 0x20AA | PMERR_INV_ORDER_LENGTH         |
| 0x20AB | PMERR_INV_ORDERING_PARM        |
| 0x20AC | PMERR_INV_OUTSIDE_DRAW_MODE    |
| 0x20AD | PMERR_INV_PAGE_VIEWPORT        |
| 0x20AE | PMERR_INV_PATH_ID              |
| 0x20AF | PMERR_INV_PATH_MODE            |
| 0x20B0 | PMERR_INV_PATTERN_ATTR         |
| 0x20B1 | PMERR_INV_PATTERN_REF_PT_ATTR  |
| 0x20B2 | PMERR_INV_PATTERN_SET_ATTR     |
| 0x20B3 | PMERR_INV_PATTERN_SET_FONT     |
| 0x20B4 | PMERR_INV_PICK_APERTURE_OPTION |
| 0x20B5 | PMERR_INV_PICK_APERTURE_POSN   |
| 0x20B6 | PMERR_INV_PICK_APERTURE_SIZE   |
| 0x20B7 | PMERR_INV_PICK_NUMBER          |
| 0x20B8 | PMERR_INV_PLAY_METAFILE_OPTION |
| 0x20B9 | PMERR_INV_PRIMITIVE_TYPE       |
| 0x20BA | PMERR_INV_PS_SIZE              |
| 0x20BB | PMERR_INV_PUTDATA_FORMAT       |
| 0x20BC | PMERR_INV_QUERY_ELEMENT_NO     |
| 0x20BD | PMERR_INV_RECT                 |
| 0x20BE | PMERR_INV_REGION_CONTROL       |
| 0x20BF | PMERR_INV_REGION_MIX_MODE      |
| 0x20C0 | PMERR_INV_REPLACE_MODE_FUNC    |
| 0x20C1 | PMERR_INV_RESERVED_FIELD       |
| 0x20C2 | PMERR_INV_RESET_OPTIONS        |
| 0x20C3 | PMERR_INV_RGBCOLOR             |
| 0x20C4 | PMERR_INV_SCAN_START           |
| 0x20C5 | PMERR_INV_SEG_ATTR             |
| 0x20C6 | PMERR_INV_SEG_ATTR_VALUE       |
| 0x20C7 | PMERR_INV_SEG_CH_LENGTH        |

|        |                                 |
|--------|---------------------------------|
| 0x20C8 | PMERR_INV_SEG_NAME              |
| 0x20C9 | PMERR_INV_SEG_OFFSET            |
| 0x20CA | PMERR_INV_SETID                 |
| 0x20CB | PMERR_INV_SETID_TYPE            |
| 0x20CC | PMERR_INV_SET_VIEWPORT_OPTION   |
| 0x20CD | PMERR_INV_SHARPNESS_PARM        |
| 0x20CE | PMERR_INV_SOURCE_OFFSET         |
| 0x20CF | PMERR_INV_STOP_DRAW_VALUE       |
| 0x20D0 | PMERR_INV_TRANSFORM_TYPE        |
| 0x20D1 | PMERR_INV_USAGE_PARM            |
| 0x20D2 | PMERR_INV_VIEWING_LIMITS        |
| 0x20D3 | PMERR_JFILE_BUSY                |
| 0x20D4 | PMERR_JNL_FUNC_DATA_TOO_LONG    |
| 0x20D5 | PMERR_KERNING_NOT_SUPPORTED     |
| 0x20D6 | PMERR_LABEL_NOT_FOUND           |
| 0x20D7 | PMERR_MATRIX_OVERFLOW           |
| 0x20D8 | PMERR_METAFILE_INTERNAL_ERROR   |
| 0x20D9 | PMERR_METAFILE_IN_USE           |
| 0x20DA | PMERR_METAFILE_LIMIT_EXCEEDED   |
| 0x20DB | PMERR_NAME_STACK_FULL           |
| 0x20DC | PMERR_NOT_CREATED_BY_DEVOPENDC  |
| 0x20DD | PMERR_NOT_IN_AREA               |
| 0x20DE | PMERR_NOT_IN_DRAW_MODE          |
| 0x20DF | PMERR_NOT_IN_ELEMENT            |
| 0x20E0 | PMERR_NOT_IN_IMAGE              |
| 0x20E1 | PMERR_NOT_IN_PATH               |
| 0x20E2 | PMERR_NOT_IN_RETAIN_MODE        |
| 0x20E3 | PMERR_NOT_IN_SEG                |
| 0x20E4 | PMERR_NO_BITMAP_SELECTED        |
| 0x20E5 | PMERR_NO_CURRENT_ELEMENT        |
| 0x20E6 | PMERR_NO_CURRENT_SEG            |
| 0x20E7 | PMERR_NO_METAFILE_RECORD_HANDLE |
| 0x20E8 | PMERR_ORDER_TOO_BIG             |
| 0x20E9 | PMERR_OTHER_SET_ID_REFS         |
| 0x20EA | PMERR_OVERRAN_SEG               |
| 0x20EB | PMERR_OWN_SET_ID_REFS           |
| 0x20EC | PMERR_PATH_INCOMPLETE           |
| 0x20ED | PMERR_PATH_LIMIT_EXCEEDED       |
| 0x20EE | PMERR_PATH_UNKNOWN              |
| 0x20EF | PMERR_PEL_IS_CLIPPED            |
| 0x20F0 | PMERR_PEL_NOT_AVAILABLE         |
| 0x20F1 | PMERR_PRIMITIVE_STACK_EMPTY     |
| 0x20F2 | PMERR_PROLOG_ERROR              |
| 0x20F3 | PMERR_PROLOG_SEG_ATTR_NOT_SET   |
| 0x20F4 | PMERR_PS_BUSY                   |
| 0x20F5 | PMERR_PS_IS_ASSOCIATED          |
| 0x20F6 | PMERR_RAM_JNL_FILE_TOO_SMALL    |
| 0x20F7 | PMERR_REALIZE_NOT_SUPPORTED     |

|        |                                 |
|--------|---------------------------------|
| 0x20F8 | PMERR_REGION_IS_CLIP_REGION     |
| 0x20F9 | PMERR_RESOURCE_DEPLETION        |
| 0x20FA | PMERR_SEG_AND_REFSEG_ARE_SAME   |
| 0x20FB | PMERR_SEG_CALL_RECURSIVE        |
| 0x20FC | PMERR_SEG_CALL_STACK_EMPTY      |
| 0x20FD | PMERR_SEG_CALL_STACK_FULL       |
| 0x20FE | PMERR_SEG_IS_CURRENT            |
| 0x20FF | PMERR_SEG_NOT_CHAINED           |
| 0x2100 | PMERR_SEG_NOT_FOUND             |
| 0x2101 | PMERR_SEG_STORE_LIMIT_EXCEEDED  |
| 0x2102 | PMERR_SETID_IN_USE              |
| 0x2103 | PMERR_SETID_NOT_FOUND           |
| 0x2104 | PMERR_STARTDOC_NOT_ISSUED       |
| 0x2105 | PMERR_STOP_DRAW_OCCURRED        |
| 0x2106 | PMERR_TOO_MANY_METAFILES_IN_USE |
| 0x2107 | PMERR_TRUNCATED_ORDER           |
| 0x2108 | PMERR_UNCHAINED_SEG_ZERO_INV    |
| 0x2109 | PMERR_UNSUPPORTED_ATTR          |
| 0x210A | PMERR_UNSUPPORTED_ATTR_VALUE    |
| 0x210B | PMERR_ENDDOC_NOT_ISSUED         |
| 0x210C | PMERR_PS_NOT_ASSOCIATED         |
| 0x210D | PMERR_INV_FLOOD_FILL_OPTIONS    |
| 0x210E | PMERR_INV_FACENAME              |
| 0x210F | PMERR_PALETTE_SELECTED          |
| 0x2110 | PMERR_NO_PALETTE_SELECTED       |
| 0x2111 | PMERR_INV_HPAL                  |
| 0x2112 | PMERR_PALETTE_BUSY              |
| 0x2113 | PMERR_START_POINT_CLIPPED       |
| 0x2114 | PMERR_NO_FILL                   |
| 0x2115 | PMERR_INV_FACENAMEDESC          |
| 0x2116 | PMERR_INV_BITMAP_DATA           |
| 0x2117 | PMERR_INV_CHAR_ALIGN_ATTR       |
| 0x2118 | PMERR_INV_HFONT                 |
| 0x2119 | PMERR_HFONT_IS_SELECTED         |
| 0x2120 | PMERR_DRVR_NOT_SUPPORTED        |
| 0x2120 | PMERR_RASTER_FONT               |
| 0x3001 | HMERR_DDF_MEMORY                |
| 0x3002 | HMERR_DDF_ALIGN_TYPE            |
| 0x3003 | HMERR_DDF_BACKCOLOR             |
| 0x3004 | HMERR_DDF_FORECOLOR             |
| 0x3005 | HMERR_DDF_FONTSTYLE             |
| 0x3006 | HMERR_DDF_REFTYPE               |
| 0x3007 | HMERR_DDF_LIST_UNCLOSED         |
| 0x3008 | HMERR_DDF_LIST_UNINITIALIZED    |
| 0x3009 | HMERR_DDF_LIST_BREAKTYPE        |
| 0x300A | HMERR_DDF_LIST_SPACING          |
| 0x300B | HMERR_DDF_HINSTANCE             |
| 0x300C | HMERR_DDF_EXCEED_MAX_LENGTH     |

|        |                                 |
|--------|---------------------------------|
| 0x300D | HMERR_DDF_EXCEED_MAX_INC        |
| 0x300E | HMERR_DDF_INVALID_DDF           |
| 0x300F | HMERR_DDF_FORMAT_TYPE           |
| 0x3010 | HMERR_DDF_INVALID_PARM          |
| 0x3011 | HMERR_DDF_INVALID_FONT          |
| 0x3012 | HMERR_DDF_SEVERE                |
| 0x4001 | PMERR_SPL_DRIVER_ERROR          |
| 0x4001 | MERR_SPL_DRIVER_ERROR           |
| 0x4002 | PMERR_SPL_DEVICE_ERROR          |
| 0x4002 | MERR_SPL_DEVICE_ERROR           |
| 0x4003 | PMERR_SPL_DEVICE_NOT_INSTALLED  |
| 0x4003 | MERR_SPL_DEVICE_NOT_INSTALLED   |
| 0x4004 | PMERR_SPL_QUEUE_ERROR           |
| 0x4004 | MERR_SPL_QUEUE_ERROR            |
| 0x4005 | PMERR_SPL_INV_HSPL              |
| 0x4005 | MERR_SPL_INV_HSPL               |
| 0x4006 | PMERR_SPL_NO_DISK_SPACE         |
| 0x4006 | MERR_SPL_NO_DISK_SPACE          |
| 0x4007 | PMERR_SPL_NO_MEMORY             |
| 0x4007 | MERR_SPL_NO_MEMORY              |
| 0x4008 | PMERR_SPL_PRINT_ABORT           |
| 0x4008 | MERR_SPL_PRINT_ABORT            |
| 0x4009 | PMERR_SPL_SPOOLER_NOT_INSTALLED |
| 0x4009 | MERR_SPL_SPOOLER_NOT_INSTALLED  |
| 0x400A | PMERR_SPL_INV_FORMS_CODE        |
| 0x400A | MERR_SPL_INV_FORMS_CODE         |
| 0x400B | PMERR_SPL_INV_PRIORITY          |
| 0x400B | MERR_SPL_INV_PRIORITY           |
| 0x400C | PMERR_SPL_NO_FREE_JOB_ID        |
| 0x400C | MERR_SPL_NO_FREE_JOB_ID         |
| 0x400D | PMERR_SPL_NO_DATA               |
| 0x400D | MERR_SPL_NO_DATA                |
| 0x400E | PMERR_SPL_INV_TOKEN             |
| 0x400E | MERR_SPL_INV_TOKEN              |
| 0x400F | PMERR_SPL_INV_DATATYPE          |
| 0x400F | MERR_SPL_INV_DATATYPE           |
| 0x4010 | PMERR_SPL_PROCESSOR_ERROR       |
| 0x4010 | MERR_SPL_PROCESSOR_ERROR        |
| 0x4011 | PMERR_SPL_INV_JOB_ID            |
| 0x4011 | MERR_SPL_INV_JOB_ID             |
| 0x4012 | PMERR_SPL_JOB_NOT_PRINTING      |
| 0x4012 | MERR_SPL_JOB_NOT_PRINTING       |
| 0x4013 | PMERR_SPL_JOB_PRINTING          |
| 0x4013 | MERR_SPL_JOB_PRINTING           |
| 0x4014 | PMERR_SPL_QUEUE_ALREADY_EXISTS  |
| 0x4014 | MERR_SPL_QUEUE_ALREADY_EXISTS   |
| 0x4015 | PMERR_SPL_INV_QUEUE_NAME        |
| 0x4015 | MERR_SPL_INV_QUEUE_NAME         |

|        |                                  |
|--------|----------------------------------|
| 0x4016 | PMERR_SPL_QUEUE_NOT_EMPTY        |
| 0x4016 | MERR_SPL_QUEUE_NOT_EMPTY         |
| 0x4017 | PMERR_SPL_DEVICE_ALREADY_EXISTS  |
| 0x4017 | MERR_SPL_DEVICE_ALREADY_EXISTS   |
| 0x4018 | PMERR_SPL_DEVICE_LIMIT_REACHED   |
| 0x4018 | MERR_SPL_DEVICE_LIMIT_REACHED    |
| 0x4019 | PMERR_SPL_STATUS_STRING_TRUNC    |
| 0x4019 | MERR_SPL_STATUS_STRING_TRUNC     |
| 0x401A | PMERR_SPL_INV_LENGTH_OR_COUNT    |
| 0x401A | MERR_SPL_INV_LENGTH_OR_COUNT     |
| 0x401B | PMERR_SPL_FILE_NOT_FOUND         |
| 0x401B | MERR_SPL_FILE_NOT_FOUND          |
| 0x401C | PMERR_SPL_CANNOT_OPEN_FILE       |
| 0x401C | MERR_SPL_CANNOT_OPEN_FILE        |
| 0x401D | PMERR_SPL_DRIVER_NOT_INSTALLED   |
| 0x401D | MERR_SPL_DRIVER_NOT_INSTALLED    |
| 0x401E | PMERR_SPL_INV_PROCESSOR_DATATYPE |
| 0x401E | MERR_SPL_INV_PROCESSOR_DATATYPE  |
| 0x401F | PMERR_SPL_INV_DRIVER_DATATYPE    |
| 0x401F | MERR_SPL_INV_DRIVER_DATATYPE     |
| 0x4020 | PMERR_SPL_PROCESSOR_NOT_INST     |
| 0x4020 | MERR_SPL_PROCESSOR_NOT_INST      |
| 0x4021 | PMERR_SPL_NO_SUCH_LOG_ADDRESS    |
| 0x4021 | MERR_SPL_NO_SUCH_LOG_ADDRESS     |
| 0x4022 | PMERR_SPL_PRINTER_NOT_FOUND      |
| 0x4022 | MERR_SPL_PRINTER_NOT_FOUND       |
| 0x4023 | PMERR_SPL_DD_NOT_FOUND           |
| 0x4023 | MERR_SPL_DD_NOT_FOUND            |
| 0x4024 | PMERR_SPL_QUEUE_NOT_FOUND        |
| 0x4024 | MERR_SPL_QUEUE_NOT_FOUND         |
| 0x4025 | PMERR_SPL_MANY_QUEUES_ASSOC      |
| 0x4025 | MERR_SPL_MANY_QUEUES_ASSOC       |
| 0x4026 | PMERR_SPL_NO_QUEUES_ASSOCIATED   |
| 0x4026 | MERR_SPL_NO_QUEUES_ASSOCIATED    |
| 0x4027 | PMERR_SPL_INI_FILE_ERROR         |
| 0x4027 | MERR_SPL_INI_FILE_ERROR          |
| 0x4028 | PMERR_SPL_NO_DEFAULT_QUEUE       |
| 0x4028 | MERR_SPL_NO_DEFAULT_QUEUE        |
| 0x4029 | PMERR_SPL_NO_CURRENT_FORMS_CODE  |
| 0x4029 | MERR_SPL_NO_CURRENT_FORMS_CODE   |
| 0x402A | PMERR_SPL_NOT_AUTHORISED         |
| 0x402A | MERR_SPL_NOT_AUTHORISED          |
| 0x402B | PMERR_SPL_TEMP_NETWORK_ERROR     |
| 0x402B | MERR_SPL_TEMP_NETWORK_ERROR      |
| 0x402C | PMERR_SPL_HARD_NETWORK_ERROR     |
| 0x402C | MERR_SPL_HARD_NETWORK_ERROR      |
| 0x402D | PMERR_DEL_NOT_ALLOWED            |
| 0x402D | MERR_DEL_NOT_ALLOWED             |



|        |                                 |
|--------|---------------------------------|
| 0x402E | PMERR_CANNOT_DEL_QP_REF         |
| 0x402E | MERR_CANNOT_DEL_QP_REF          |
| 0x402F | PMERR_CANNOT_DEL_QNAME_REF      |
| 0x402F | MERR_CANNOT_DEL_QNAME_REF       |
| 0x4030 | PMERR_CANNOT_DEL_PRINTER_DD_REF |
| 0x4030 | MERR_CANNOT_DEL_PRINTER_DD_REF  |
| 0x4031 | PMERR_CANNOT_DEL_PRN_NAME_REF   |
| 0x4031 | MERR_CANNOT_DEL_PRN_NAME_REF    |
| 0x4032 | PMERR_CANNOT_DEL_PRN_ADDR_REF   |
| 0x4032 | MERR_CANNOT_DEL_PRN_ADDR_REF    |
| 0x4033 | PMERR_SPOOLER_QP_NOT_DEFINED    |
| 0x4033 | MERR_SPOOLER_QP_NOT_DEFINED     |
| 0x4034 | PMERR_PRN_NAME_NOT_DEFINED      |
| 0x4034 | MERR_PRN_NAME_NOT_DEFINED       |
| 0x4035 | PMERR_PRN_ADDR_NOT_DEFINED      |
| 0x4035 | MERR_PRN_ADDR_NOT_DEFINED       |
| 0x4036 | PMERR_PRINTER_DD_NOT_DEFINED    |
| 0x4036 | MERR_PRINTER_DD_NOT_DEFINED     |
| 0x4037 | PMERR_PRINTER_QUEUE_NOT_DEFINED |
| 0x4037 | MERR_PRINTER_QUEUE_NOT_DEFINED  |
| 0x4038 | PMERR_PRN_ADDR_IN_USE           |
| 0x4038 | MERR_PRN_ADDR_IN_USE            |
| 0x4039 | PMERR_SPL_TOO_MANY_OPEN_FILES   |
| 0x4039 | MERR_SPL_TOO_MANY_OPEN_FILES    |
| 0x403A | PMERR_SPL_CP_NOT_REQD           |
| 0x403A | MERR_SPL_CP_NOT_REQD            |
| 0x4040 | PMERR_UNABLE_TO_CLOSE_DEVICE    |
| 0x4040 | MERR_UNABLE_TO_CLOSE_DEVICE     |
| 0x4FC9 | PMERR_SPLMSGBOX_INFO_CAPTION    |
| 0x4FC9 | MERR_SPLMSGBOX_INFO_CAPTION     |
| 0x4FCA | PMERR_SPLMSGBOX_WARNING_CAPTION |
| 0x4FCA | MERR_SPLMSGBOX_WARNING_CAPTION  |
| 0x4FCB | PMERR_SPLMSGBOX_ERROR_CAPTION   |
| 0x4FCB | MERR_SPLMSGBOX_ERROR_CAPTION    |
| 0x4FCC | PMERR_SPLMSGBOX_SEVERE_CAPTION  |
| 0x4FCC | MERR_SPLMSGBOX_SEVERE_CAPTION   |
| 0x4FCD | PMERR_SPLMSGBOX_JOB_DETAILS     |
| 0x4FCD | MERR_SPLMSGBOX_JOB_DETAILS      |
| 0x4FCE | PMERR_SPLMSGBOX_ERROR_ACTION    |
| 0x4FCE | MERR_SPLMSGBOX_ERROR_ACTION     |
| 0x4FCF | PMERR_SPLMSGBOX_SEVERE_ACTION   |
| 0x4FCF | MERR_SPLMSGBOX_SEVERE_ACTION    |
| 0x4FD0 | PMERR_SPLMSGBOX_BIT_0_TEXT      |
| 0x4FD0 | MERR_SPLMSGBOX_BIT_0_TEXT       |
| 0x4FD1 | PMERR_SPLMSGBOX_BIT_1_TEXT      |
| 0x4FD1 | MERR_SPLMSGBOX_BIT_1_TEXT       |
| 0x4FD2 | PMERR_SPLMSGBOX_BIT_2_TEXT      |
| 0x4FD2 | MERR_SPLMSGBOX_BIT_2_TEXT       |

|                  |                              |
|------------------|------------------------------|
| 0x4FD3           | PMERR_SPLMSGBOX_BIT_3_TEXT   |
| 0x4FD3           | MERR_SPLMSGBOX_BIT_3_TEXT    |
| 0x4FD4           | PMERR_SPLMSGBOX_BIT_4_TEXT   |
| 0x4FD4           | MERR_SPLMSGBOX_BIT_4_TEXT    |
| 0x4FD5           | PMERR_SPLMSGBOX_BIT_5_TEXT   |
| 0x4FD5           | MERR_SPLMSGBOX_BIT_5_TEXT    |
| 0x4FD6           | PMERR_SPLMSGBOX_BIT_15_TEXT  |
| 0x4FD6           | MERR_SPLMSGBOX_BIT_15_TEXT   |
| 0x4FD7           | PMERR_SPL_NOPATHBUFFER       |
| 0x4FD7           | MERR_SPL_NOPATHBUFFER        |
| 0x4FD8           | MERR_SPL_ALREADY_INITIALISED |
| 0x4FD9           | MERR_SPL_ERROR               |
| 0x5001           | PMERR_INV_TYPE               |
| 0x5001           | MERR_INV_TYPE                |
| 0x5002           | PMERR_INV_CONV               |
| 0x5002           | MERR_INV_CONV                |
| 0x5003           | PMERR_INV_SEGLEN             |
| 0x5003           | MERR_INV_SEGLEN              |
| 0x5004           | PMERR_DUP_SEGNAME            |
| 0x5004           | MERR_DUP_SEGNAME             |
| 0x5005           | PMERR_INV_XFORM              |
| 0x5005           | MERR_INV_XFORM               |
| 0x5006           | PMERR_INV_VIEWLIM            |
| 0x5006           | MERR_INV_VIEWLIM             |
| 0x5007           | PMERR_INV_3DCOORD            |
| 0x5007           | MERR_INV_3DCOORD             |
| 0x5008           | PMERR_SMB_OVFLOW             |
| 0x5008           | MERR_SMB_OVFLOW              |
| 0x5009           | PMERR_SEG_OVFLOW             |
| 0x5009           | MERR_SEG_OVFLOW              |
| 0x5010           | PMERR_PIC_DUP_FILENAME       |
| 0x5010           | MERR_PIC_DUP_FILENAME        |
| SPLERR_BASE+0FA1 | PMERR_SPL_ERROR_1            |
| SPLERR_BASE+0FA2 | PMERR_SPL_ERROR_2            |
| SPLERR_BASE+0FA3 | PMERR_SPL_ERROR_3            |
| SPLERR_BASE+0FA4 | PMERR_SPL_ERROR_4            |
| SPLERR_BASE+0FA5 | PMERR_SPL_ERROR_5            |
| SPLERR_BASE+0FA6 | PMERR_SPL_ERROR_6            |
| SPLERR_BASE+0FA7 | PMERR_SPL_ERROR_7            |
| SPLERR_BASE+0FA8 | PMERR_SPL_ERROR_8            |
| SPLERR_BASE+0FA9 | PMERR_SPL_ERROR_9            |
| SPLERR_BASE+0FAA | PMERR_SPL_ERROR_10           |
| SPLERR_BASE+0FAB | PMERR_SPL_ERROR_11           |
| SPLERR_BASE+0FAC | PMERR_SPL_ERROR_12           |
| SPLERR_BASE+0FAD | PMERR_SPL_ERROR_13           |
| SPLERR_BASE+0FAE | PMERR_SPL_ERROR_14           |
| SPLERR_BASE+0FAF | PMERR_SPL_ERROR_15           |
| SPLERR_BASE+0FB0 | PMERR_SPL_ERROR_16           |

|                  |                               |
|------------------|-------------------------------|
| SPLERR_BASE+0FB1 | PMERR_SPL_ERROR_17            |
| SPLERR_BASE+0FB2 | PMERR_SPL_ERROR_18            |
| SPLERR_BASE+0FB3 | PMERR_SPL_ERROR_19            |
| SPLERR_BASE+0FB4 | PMERR_SPL_ERROR_20            |
| SPLERR_BASE+0FB5 | PMERR_SPL_ERROR_21            |
| SPLERR_BASE+0FB6 | PMERR_SPL_ERROR_22            |
| SPLERR_BASE+0FB7 | PMERR_SPL_ERROR_23            |
| SPLERR_BASE+0FB8 | PMERR_SPL_ERROR_24            |
| SPLERR_BASE+0FB9 | PMERR_SPL_ERROR_25            |
| SPLERR_BASE+0FBA | PMERR_SPL_ERROR_26            |
| SPLERR_BASE+0FBB | PMERR_SPL_ERROR_27            |
| SPLERR_BASE+0FBC | PMERR_SPL_ERROR_28            |
| SPLERR_BASE+0FBD | PMERR_SPL_ERROR_29            |
| SPLERR_BASE+0FBE | PMERR_SPL_ERROR_30            |
| SPLERR_BASE+0FBF | PMERR_SPL_ERROR_31            |
| SPLERR_BASE+0FC0 | PMERR_SPL_ERROR_32            |
| SPLERR_BASE+0FC1 | PMERR_SPL_ERROR_33            |
| SPLERR_BASE+0FC2 | PMERR_SPL_ERROR_34            |
| SPLERR_BASE+0FC3 | PMERR_SPL_ERROR_35            |
| SPLERR_BASE+0FC4 | PMERR_SPL_ERROR_36            |
| SPLERR_BASE+0FC5 | PMERR_SPL_ERROR_37            |
| SPLERR_BASE+0FC6 | PMERR_SPL_ERROR_38            |
| SPLERR_BASE+0FC7 | PMERR_SPL_ERROR_39            |
| SPLERR_BASE+0FC8 | PMERR_SPL_ERROR_40            |
| SPLERR_BASE+0FFF | PMERR_SPL_ERROR               |
| SPLERR_BASE+0FFD | PMERR_SPL_ALREADY_INITIALISED |

---

## Appendix C. Error Explanations

This appendix gives an explanation for each PM error. The errors are listed in alphabetic order. The number associated with each error is given in Appendix B, "Error Codes" on page B-1.

| <b>Error Constant</b>               | <b>Explanation</b>   |
|-------------------------------------|--|
| <b>HMERR_ALLOCATE_SEGMENT</b>       | Unable to allocate a segment of memory for memory allocation requests from the Help Manager. |
| <b>HMERR_CLOSE_LIB_FILE</b>         | The library file cannot be closed.   |
| <b>HMERR_CONTENT_NOT_FOUND</b>      | The library file does not have any content.  |
| <b>HMERR_DATABASE_NOT_OPEN</b>      | Unable to read the unopened database.  |
| <b>HMERR_DDF_ALIGN_TYPE</b>         | The alignment type is not valid.   |
| <b>HMERR_DDF_BACKCOLOR</b>          | The background color is not valid.   |
| <b>HMERR_DDF_EXCEED_MAX_INC</b>     | The value specified to increment DDF memory is too large.                                    |
| <b>HMERR_DDF_EXCEED_MAX_LENGTH</b>  | The amount of data is too large for the DDF buffer.  |
| <b>HMERR_DDF_FONTSTYLE</b>          | The font style is not valid.   |
| <b>HMERR_DDF_FORECOLOR</b>          | The foreground color is not valid.   |
| <b>HMERR_DDF_FORMAT_TYPE</b>        | The format type specified is invalid.  |
| <b>HMERR_DDF_HINSTANCE</b>          | The DDF instance is invalid.   |
| <b>HMERR_DDF_INVALID_DDF</b>        | The DDF handle is invalid.   |
| <b>HMERR_DDF_INVALID_FONT</b>       | The font value specified is invalid.   |
| <b>HMERR_DDF_INVALID_PARM</b>       | One of the DDF parameters specified is invalid.  |
| <b>HMERR_DDF_LIST_BREAKTYPE</b>     | The value of BreakType is not valid.   |
| <b>HMERR_DDF_LIST_SPACING</b>       | The value for Spacing is not valid.  |
| <b>HMERR_DDF_LIST_UNCLOSED</b>      | An attempt was made to nest a list.  |
| <b>HMERR_DDF_LIST_UNINITIALIZED</b> | No definition list has been initialized by DdfBeginList.                                     |
| <b>HMERR_DDF_MEMORY</b>             | Not enough memory is available.  |
| <b>HMERR_DDF_REFTYPE</b>            | The reference type is not valid.   |
| <b>HMERR_DDF_SEVERE</b>             | Internal error detected by the Help Manager.   |
| <b>HMERR_FREE_MEMORY</b>            | Unable to free allocated memory.   |

|  |   |
|--|---|
| <b>HMERR_HELP_INST_CALLED_INVALID</b>  | The handle of the instance specified on a call to the Help Manager does not have the class name of a Help Manager instance. |
| <b>HMERR_HELP_INSTANCE_UNDEFINE</b>    | The help instance handle specified is invalid.  |
| <b>HMERR_HELPITEM_NOT_FOUND</b>        | Context-sensitive help was requested but the ID of the main help item specified was not found in the help table.            |
| <b>HMERR_HELPSUBITEM_NOT_FOUND</b>     | Context-sensitive help was requested but the ID of the help item specified was not found in the help subtable.              |
| <b>HMERR_HELPTABLE_UNDEFINE</b>        | The application did not provide a help table for context-sensitive help.  |
| <b>HMERR_INDEX_NOT_FOUND</b>           | The index is not in the library file.   |
| <b>HMERR_INVALID_ASSOC_APP_WND</b>     | The application window handle specified on the WinAssociateHelpInstance function is not a valid window handle.              |
| <b>HMERR_INVALID_ASSOC_HELP_INST</b>   | The help instance handle specified on the WinAssociateHelpInstance function is not a valid window handle.                   |
| <b>HMERR_INVALID_DESTROY_HELP_INST</b> | The window handle specified as the help instance to destroy is not of the help instance class.                              |
| <b>HMERR_INVALID_HELP_INSTANCE_HDL</b> | The handle specified to be a help instance does not have the class name of a Help Manager instance.                         |
| <b>HMERR_INVALID_HELPSUBITEM_SIZE</b>  | The help subtable item size is less than 2.   |
| <b>HMERR_INVALID_LIB_FILE</b>          | Improper library file provided.   |
| <b>HMERR_INVALID_QUERY_APP_WND</b>     | The application window specified on a WinQueryHelpInstance function is not a valid window handle.                           |
| <b>HMERR_LOAD_DLL</b>                  | Unable to load resource data link library.  |
| <b>HMERR_NO_FRAME_WND_IN_CHAIN</b>     | There is no frame window in the window chain from which to find or set the associated help instance.                        |
| <b>HMERR_NO_HELP_INST_IN_CHAIN</b>     | The parent or owner chain of the application window specified does not have an associated help instance.                    |
| <b>HMERR_NO_MEMORY</b>                 | Unable to allocate the requested amount of memory.  |

|                                       |  |
|---------------------------------------|--|
| <b>HMERR_OPEN_LIB_FILE</b>            | The library file cannot be opened.   |
| <b>HMERR_PANEL_NOT_FOUND</b>          | Unable to find the requested help panel.   |
| <b>HMERR_READ_LIB_FILE</b>            | The library file cannot be read.   |
| <b>PMERR_ACCESS_DENIED</b>            | The memory block was not allocated properly.   |
| <b>PMERR_ALREADY_IN_AREA</b>          | An attempt was made to begin a new area while an existing area bracket was already open.   |
| <b>PMERR_ALREADY_IN_ELEMENT</b>       | An attempt was made to begin a new element while an existing element bracket was already open.   |
| <b>PMERR_ALREADY_IN_PATH</b>          | An attempt was made to begin a new path while an existing path bracket was already open.   |
| <b>PMERR_ALREADY_IN_SEG</b>           | An attempt was made to open a new segment while an existing segment bracket was already open.  |
| <b>PMERR_APPL_STRUCTURE_TOO_SMALL</b> | The application buffer length is less than the total length required for the (application) component types.  |
| <b>PMERR_AREA_INCOMPLETE</b>          | One of the following has occurred: <ul style="list-style-type: none"> <li>• A segment has been opened, closed, or drawn.</li> <li>• GpiAssociate was issued while an area bracket was open.</li> <li>• A drawn segment has opened an area bracket and ended without closing it.</li> </ul> |
| <b>PMERR_ARRAY_TOO_LARGE</b>          | More than 4 bytes was attempted to be inserted or extracted.   |
| <b>PMERR_ARRAY_TOO_SMALL</b>          | The array specified was too small.   |
| <b>PMERR_ATOM_NAME_NOT_FOUND</b>      | The specified atom name is not in the atom table.  |
| <b>PMERR_BASE_ERROR</b>               | An OS/2 base error has occurred. The base error code can be accessed using the OffBinaryData field of the ERRINFO structure returned by WinGetErrorInfo.   |
| <b>PMERR_BITMAP_IN_USE</b>            | An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier  |

|  |  |
|--|--|
| <b>PMERR_BITMAP_IS_SELECTED</b>        | (setid) using GpiSetBitmapId while it was already tagged with an existing setid.   |
| <b>PMERR_BITMAP_NOT_FOUND</b>          | An attempt was made to delete a bit map while it was selected into a device context.   |
| <b>PMERR_BITMAP_NOT_SELECTED</b>       | A attempt was made to perform a bit-map operation on a bit map that did not exist.   |
| <b>PMERR_BOUNDS_OVERFLOW</b>           | A attempt was made to perform an operation on presentation space associated with a memory device context that had no selected bit map.                                     |
| <b>PMERR_BUFFER_TOO_SMALL</b>          | An internal overflow error occurred during boundary data accumulation. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large. |
| <b>PMERR_C_LENGTH_TOO_SMALL</b>        | The supplied buffer was not large enough for the data to be returned.  |
| <b>PMERR_C_LENGTH_TOO_SMALL</b>        | The maximum length of the C structure is less than the total length required for the (C) component types.  |
| <b>PMERR_CALLED_SEG_IS_CHAINED</b>     | An attempt was made to call a segment that has a chained attribute set.  |
| <b>PMERR_CALLED_SEG_IS_CURRENT</b>     | An attempt was made to call a segment that is currently open.  |
| <b>PMERR_CALLED_SEG_NOT_FOUND</b>      | An attempt was made to call a segment that did not exist.  |
| <b>PMERR_CAN_NOT_CALL_SPOOLER</b>      | An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.   |
| <b>PMERR_CANNOT_DEL_PRINTER_DD_REF</b> | Presentation Manager device driver deletion not possible due to a reference.   |
| <b>PMERR_CANNOT_DEL_PRN_ADDR_REF</b>   | Printer port deletion not possible due to a reference.   |
| <b>PMERR_CANNOT_DEL_PRN_NAME_REF</b>   | Printer deletion not possible due to a reference.  |
| <b>PMERR_CANNOT_DEL_QNAME_REF</b>      | Spooler queue deletion not possible due to a reference.  |
| <b>PMERR_CANNOT_DEL_QP_REF</b>         | Spooler queue processor deletion not possible due to a reference.  |
| <b>PMERR_CANNOT_STOP</b>               | The session cannot be stopped.   |

|                                       |  |
|---------------------------------------|--|
| <b>PMERR_COL_TABLE_NOT_REALIZABLE</b> | An attempt was made to realize a color table that is not realizable.   |
| <b>PMERR_COL_TABLE_NOT_REALIZED</b>   | An attempt was made to realize a color table on a device driver that does not support this function.   |
| <b>PMERR_COORDINATE_OVERFLOW</b>      | An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.                    |
| <b>PMERR_DATA_TOO_LONG</b>            | An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.                              |
| <b>PMERR_DATATYPE_ENTRY_BAD_INDEX</b> | An invalid datatype entry index was specified.   |
| <b>PMERR_DATATYPE_ENTRY_CTL_BAD</b>   | An invalid datatype entry control was specified.   |
| <b>PMERR_DATATYPE_ENTRY_CTL_MISS</b>  | The datatype entry control was missing.  |
| <b>PMERR_DATATYPE_ENTRY_NOT_NUM</b>   | The datatype entry specified was not numerical.  |
| <b>PMERR_DATATYPE_ENTRY_NOT_OFF</b>   | The datatype entry specified was not an offset.  |
| <b>PMERR_DATATYPE_INVALID</b>         | An invalid datatype was specified.   |
| <b>PMERR_DATATYPE_NOT_UNIQUE</b>      | An attempt to register a datatype failed because it is not unique.   |
| <b>PMERR_DATATYPE_TOO_LONG</b>        | The datatype specified was too long.   |
| <b>PMERR_DATATYPE_TOO_SMALL</b>       | The datatype specified was too small.  |
| <b>PMERR_DC_IS_ASSOCIATED</b>         | An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.            |
| <b>PMERR_DEL_NOT_ALLOWED</b>          | Deletion not possible.   |
| <b>PMERR_DESC_STRING_TRUNCATED</b>    | An attempt was made to supply a description string with GpiBeginElement that was greater than the permitted maximum length (251 characters). The string was truncated. |
| <b>PMERR_DEV_FUNC_NOT_INSTALLED</b>   | The function requested is not supported by the presentation driver.  |



|                                     |   |
|-------------------------------------|---|
| <b>PMERR_DEVICE_DRIVER_ERROR_1</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_10</b> | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_2</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_3</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_4</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_5</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_6</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_7</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_8</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DEVICE_DRIVER_ERROR_9</b>  | Miscellaneous error available for use by user written device drivers.   |
| <b>PMERR_DOS_ERROR</b>              | A DOS call returned an error.   |
| <b>PMERR_DOSOPEN_FAILURE</b>        | A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.   |
| <b>PMERR_DOSREAD_FAILURE</b>        | A DosRead call made during GpiLoadMetaFile gave a good return code. However, it failed to read any more bytes although the file length indicated that there were more to be read.   |
| <b>PMERR_DRIVER_NOT_FOUND</b>       | The device driver specified with DevPostDeviceModes was not found.  |
| <b>PMERR_DUP_SEG</b>                | During GpiPlayMetaFile, while the actual drawing mode was <b>draw-and-retain</b> or <b>retain</b> , a metafile segment to be stored in the presentation space was found to have the same segment identifier as an existing segment. |
| <b>PMERR_DUP_SEGNAME</b>            | A called segment has a name that has already been used by another called segment in the input PIF.  |

|                                     |  |
|-------------------------------------|--|
| <b>PMERR_DUPLICATE_TITLE</b>        | The program title specified in the PIBSTRUCT already exists within the same group.   |
| <b>PMERR_DYNAMIC_SEG_SEQ_ERROR</b>  | During removal of dynamic segments while processing GpiDrawChain, GpiDrawFrom, or GpiDrawSegment, the internal state indicated that dynamic segment data was still visible after all chained dynamic segments had been processed. This can occur if segments drawn dynamically (including called segments) are modified or removed from the chain while visible. |
| <b>PMERR_DYNAMIC_SEG_ZERO_INV</b>   | An attempt was been made to open a dynamic segment with a segment identifier of zero.  |
| <b>PMERR_ENDDOC_NOT_ISSUED</b>      | A request to close the spooled output without first issuing a an ENDDOC was attempted.   |
| <b>PMERR_ESC_CODE_NOT_SUPPORTED</b> | The code specified with DevEscape is not supported by the target device driver.  |
| <b>PMERR_EXCEEDS_MAX_SEG_LENGTH</b> | During metafile creation or generation of retained graphics the system has exceeded maximum segment size.  |
| <b>PMERR_FONT_AND_MODE_MISMATCH</b> | An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.   |
| <b>PMERR_FONT_FILE_NOT_LOADED</b>   | An attempt was made to unload a font file that was not loaded.   |
| <b>PMERR_FONT_NOT_LOADED</b>        | An attempt was made to create a font that was not loaded.  |
| <b>PMERR_FUNCTION_NOT_SUPPORTED</b> | The function is not supported.   |
| <b>PMERR_GREATER_THAN_64K</b>       | A data item or array dimension is greater than 65 535.   |
| <b>PMERR_HBITMAP_BUSY</b>           | An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.  |

|                                       |   |
|---------------------------------------|---|
| <b>PMERR_HDC_BUSY</b>                 | An internal device context busy error was detected. The device context was locked by one thread during an attempt to access it from another thread.   |
| <b>PMERR_HEAP_MAX_SIZE_REACHED</b>    | The heap has reached its maximum size (64KB), and cannot be increased.  |
| <b>PMERR_HEAP_OUT_OF_MEMORY</b>       | An attempt to increase the size of the heap failed.   |
| <b>PMERR_HFONT_IS_SELECTED</b>        | An attempt has been made to either change the owner of a font, or delete when it is currently selected.   |
| <b>PMERR_HRGN_BUSY</b>                | An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.   |
| <b>PMERR_HUGE_FONTS_NOT_SUPPORTED</b> | An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.  |
| <b>PMERR_ID_HAS_NO_BITMAP</b>         | No bit map was tagged with the setid specified on a GpiQueryBitmapHandle function.  |
| <b>PMERR_IMAGE_INCOMPLETE</b>         | A drawn segment has opened an image bracket and ended without closing it.   |
| <b>PMERR_INCOMPATIBLE_BITMAP</b>      | An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.   |
| <b>PMERR_INCOMPATIBLE_METAFILE</b>    | An attempt was made to associate a presentation space and a metafile device context with incompatible page units, size or coordinate format; or to play a metafile using the RES_RESET option (to reset the presentation space) to a presentation space that is itself associated with a metafile device context. |
| <b>PMERR_INCORRECT_DATATYPE</b>       | A data type is specified which is incorrect for this function.  |
| <b>PMERR_INCORRECT_DC_TYPE</b>        | An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type   |

**PMERR\_INCORRECT\_HSTRUCT**

that is unable to support bit-map operations.

A structure handle is non-NULL, and is invalid for one of the following reasons:

- It is not the handle of a data structure.
- It is the handle of an ERRINFO structure, which should not be used in this call.
- A handle block returned by the bindings to the application has been used for an in-line structure handle.

**PMERR\_INI\_FILE\_IS\_SYS\_OR\_USER**

User or system initialization file cannot be closed.

**PMERR\_INSUFF\_SPACE\_TO\_ADD**

The initialization file could not be extended to add the required program or group.

**PMERR\_INSUFFICIENT\_DISK\_SPACE**

The operation terminated through insufficient disk space.

**PMERR\_INSUFFICIENT\_MEMORY**

The operation terminated through insufficient memory.

**PMERR\_INV\_ANGLE\_PARM**

An invalid angle parameter was specified with GpiPartialArc.

**PMERR\_INV\_ARC\_CONTROL**

An invalid control parameter was specified with GpiFullArc.

**PMERR\_INV\_AREA\_CONTROL**

An invalid options parameter was specified with GpiBeginArea.

**PMERR\_INV\_ATTR\_MODE**

An invalid mode parameter was specified with GpiSetAttrMode.

**PMERR\_INV\_BACKGROUND\_COL\_ATTR**

An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

**PMERR\_INV\_BACKGROUND\_MIX\_ATTR**

An invalid background mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

**PMERR\_INV\_BITBLT\_MIX**

An invalid *IRop* was specified with a GpiBitBlt or GpiWCBitBlt function.

**PMERR\_INV\_BITBLT\_STYLE**

An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

**PMERR\_INV\_BITMAP\_DATA**

In processing a bit map, the end of the data was unexpectedly encountered.

|                                      |   |
|--------------------------------------|---|
| <b>PMERR_INV_BITMAP_DIMENSION</b>    | An invalid dimension was specified with a load bit-map function.  |
| <b>PMERR_INV_BOX_CONTROL</b>         | An invalid control parameter was specified with GpiBox.   |
| <b>PMERR_INV_BOX_ROUNDING_PARM</b>   | An invalid corner rounding control parameter was specified with GpiBox.   |
| <b>PMERR_INV_CHAR_ALIGN_ATTR</b>     | The text alignment attribute specified in GpiSetTextAlignment is not valid.   |
| <b>PMERR_INV_CHAR_ANGLE_ATTR</b>     | The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.                                       |
| <b>PMERR_INV_CHAR_DIRECTION_ATTR</b> | An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask. |
| <b>PMERR_INV_CHAR_MODE_ATTR</b>      | An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.      |
| <b>PMERR_INV_CHAR_POS_OPTIONS</b>    | An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.   |
| <b>PMERR_INV_CHAR_SET_ATTR</b>       | An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.     |
| <b>PMERR_INV_CHAR_SHEAR_ATTR</b>     | An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.     |
| <b>PMERR_INV_CLIP_PATH_OPTIONS</b>   | An invalid options parameter was specified with GpiSetClipPath.   |
| <b>PMERR_INV_CODEPAGE</b>            | An invalid code-page parameter was specified with GpiSetCp.   |
| <b>PMERR_INV_COLOR_ATTR</b>          | An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.               |
| <b>PMERR_INV_COLOR_DATA</b>          | Invalid color table definition data was specified with GpiCreateLogColorTable.  |
| <b>PMERR_INV_COLOR_FORMAT</b>        | An invalid format parameter was specified with GpiCreateLogColorTable.  |

|                                     |  |
|-------------------------------------|--|
| <b>PMERR_INV_COLOR_INDEX</b>        | An invalid color index parameter was specified with GpiQueryRGBColor.  |
| <b>PMERR_INV_COLOR_OPTIONS</b>      | An invalid options parameter was specified with a logical color table or color query function.   |
| <b>PMERR_INV_COLOR_START_INDEX</b>  | An invalid starting index parameter was specified with a logical color table or color query function.                                      |
| <b>PMERR_INV_CONV</b>               | Invalid conversion-type parameter.   |
| <b>PMERR_INV_COORD_OFFSET</b>       | An invalid coordinate offset value was specified.  |
| <b>PMERR_INV_COORD_SPACE</b>        | An invalid source or target coordinate space parameter was specified with GpiConvert.  |
| <b>PMERR_INV_COORDINATE</b>         | An invalid coordinate value was specified.   |
| <b>PMERR_INV_CORRELATE_DEPTH</b>    | An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.                              |
| <b>PMERR_INV_CORRELATE_TYPE</b>     | An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.                                  |
| <b>PMERR_INV_CURSOR_BITMAP</b>      | An invalid pointer was referenced with WinSetPointer.  |
| <b>PMERR_INV_DC_DATA</b>            | An invalid data parameter was specified with DevOpenDC.  |
| <b>PMERR_INV_DC_TYPE</b>            | An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context. |
| <b>PMERR_INV_DEV_MODES_OPTIONS</b>  | An invalid options parameter was specified with DevPostDeviceModes.  |
| <b>PMERR_INV_DEVICE_NAME</b>        | An invalid devicename parameter was specified with DevPostDeviceModes.   |
| <b>PMERR_INV_DRAW_BORDER_OPTION</b> | An invalid option parameter was specified with WinDrawBorder.  |
| <b>PMERR_INV_DRAW_CONTROL</b>       | An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.  |
| <b>PMERR_INV_DRAW_VALUE</b>         | An invalid value parameter was specified with GpiSetDrawControl.   |

|                                     |  |
|-------------------------------------|--|
| <b>PMERR_INV_DRAWING_MODE</b>       | An invalid mode parameter was specified with GpiSetDrawControl not <b>draw-and-retain</b> or <b>draw</b> .                                       |
| <b>PMERR_INV_DRIVER_DATA</b>        | Invalid driver data was specified.   |
| <b>PMERR_INV_DRIVER_NAME</b>        | A driver name was specified which has not been installed.  |
| <b>PMERR_INV_EDIT_MODE</b>          | An invalid mode parameter was specified with GpiSetEditMode.   |
| <b>PMERR_INV_ELEMENT_OFFSET</b>     | An invalid off (offset) parameter was specified with GpiQueryElement.  |
| <b>PMERR_INV_ELEMENT_POINTER</b>    | An attempt was made to issue GpiPutData with the element pointer not pointing at the last element.   |
| <b>PMERR_INV_END_PATH_OPTIONS</b>   | An attempt to create or delete a path out of context of the path bracket was made.   |
| <b>PMERR_INV_ESC_CODE</b>           | An invalid escape code was used in a call to DevEscape.  |
| <b>PMERR_INV_ESCAPE_DATA</b>        | An invalid data parameter was specified with DevEscape.  |
| <b>PMERR_INV_FACENAME</b>           | An invalid font family name was passed to GpiQueryFaceString.  |
| <b>PMERR_INV_FACENAMEDESC</b>       | The font facename description is invalid.  |
| <b>PMERR_INV_FILL_PATH_OPTIONS</b>  | An invalid options parameter was specified with GpiFillPath.   |
| <b>PMERR_INV_FIRST_CHAR</b>         | An invalid firstchar parameter was specified with GpiQueryWidthTable.  |
| <b>PMERR_INV_FLOOD_FILL_OPTIONS</b> | Invalid flood fill parameters were specified.  |
| <b>PMERR_INV_FONT_ATTRS</b>         | An invalid attrs parameter was specified with GpiCreateLogFont.  |
| <b>PMERR_INV_FONT_FILE_DATA</b>     | The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data. |
| <b>PMERR_INV_FOR_THIS_DC_TYPE</b>   | An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.        |
| <b>PMERR_INV_FORMS_CODE</b>         | An invalid forms code parameter was specified with DevQueryHardcopyCaps.   |

|                                       |   |
|---------------------------------------|---|
| <b>PMERR_INV_GEOM_LINE_WIDTH_ATTR</b> | An invalid geometric line width attribute value was specified.  |
| <b>PMERR_INV_GETDATA_CONTROL</b>      | An invalid format parameter was specified with GpiGetData.  |
| <b>PMERR_INV_GRAPHICS_FIELD</b>       | An invalid field parameter was specified with GpiSetGraphicsField.  |
| <b>PMERR_INV_HBITMAP</b>              | An invalid bit-map handle was specified.  |
| <b>PMERR_INV_HDC</b>                  | An invalid device-context handle or (micro presentation space) presentation-space handle was specified.   |
| <b>PMERR_INV_HFONT</b>                | An invalid font handle was specified.   |
| <b>PMERR_INV_HMF</b>                  | An invalid metafile handle was specified.   |
| <b>PMERR_INV_HPAL</b>                 | An invalid color palette handle was specified.  |
| <b>PMERR_INV_HPS</b>                  | An invalid presentation-space handle was specified.   |
| <b>PMERR_INV_HRGN</b>                 | An invalid region handle was specified.   |
| <b>PMERR_INV_ID</b>                   | An invalid <i>IPSid</i> parameter was specified with GpiRestorePS.  |
| <b>PMERR_INV_IMAGE_DATA_LENGTH</b>    | An invalid <i>Length</i> parameter was specified with Gpimage. There is a mismatch between the image size and the data length.  |
| <b>PMERR_INV_IMAGE_DIMENSION</b>      | An invalid <i>pszImageSize</i> parameter was specified with Gpimage.  |
| <b>PMERR_INV_IMAGE_FORMAT</b>         | An invalid <i>Format</i> parameter was specified with Gpimage.  |
| <b>PMERR_INV_IN_AREA</b>              | An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is <b>draw</b> or <b>draw-and-retain</b> or during segment drawing or correlation functions. |
| <b>PMERR_INV_IN_CURRENT_EDIT_MODE</b> | An attempt was made to issue a function invalid inside the current editing mode.  |
| <b>PMERR_INV_IN_ELEMENT</b>           | An attempt was made to issue a function invalid inside an element bracket.  |
| <b>PMERR_INV_IN_IMAGE</b>             | An attempt was made to issue a function invalid inside an image bracket.  |
| <b>PMERR_INV_IN_PATH</b>              | An attempt was made to issue a function invalid inside a path bracket.  |



|                                     |   |
|-------------------------------------|---|
| <b>PMERR_INV_IN_RETAIN_MODE</b>     | An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not <b>draw</b> or <b>draw-and-retain</b> .  |
| <b>PMERR_INV_IN_SEG</b>             | An attempt was made to issue a function invalid inside a segment bracket.   |
| <b>PMERR_INV_IN_VECTOR_SYMBOL</b>   | An invalid order was detected inside a vector symbol definition while drawing a vector (outline) font.  |
| <b>PMERR_INV_INFO_TABLE</b>         | An invalid bit-map info table was specified with a bit-map operation.   |
| <b>PMERR_INV_LENGTH_OR_COUNT</b>    | An invalid length or count parameter was specified.   |
| <b>PMERR_INV_LINE_END_ATTR</b>      | An invalid line end attribute value was specified.  |
| <b>PMERR_INV_LINE_JOIN_ATTR</b>     | An invalid line join attribute value was specified.   |
| <b>PMERR_INV_LINE_TYPE_ATTR</b>     | An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.     |
| <b>PMERR_INV_LINE_WIDTH_ATTR</b>    | An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.    |
| <b>PMERR_INV_LOGICAL_ADDRESS</b>    | An invalid device logical address was specified.  |
| <b>PMERR_INV_MARKER_BOX_ATTR</b>    | An invalid marker box attribute value was specified.  |
| <b>PMERR_INV_MARKER_SET_ATTR</b>    | An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.    |
| <b>PMERR_INV_MARKER_SYMBOL_ATTR</b> | An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask. |
| <b>PMERR_INV_MATRIX_ELEMENT</b>     | An invalid transformation matrix element was specified.   |
| <b>PMERR_INV_MAX_HITS</b>           | An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.  |

|                                       |   |
|---------------------------------------|---|
| <b>PMERR_INV_METAFILE</b>             | An invalid metafile was specified with GpiPlayMetaFile.   |
| <b>PMERR_INV_METAFILE_LENGTH</b>      | An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.  |
| <b>PMERR_INV_METAFILE_OFFSET</b>      | An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.  |
| <b>PMERR_INV_MICROPS_DRAW_CONTROL</b> | A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.                                    |
| <b>PMERR_INV_MICROPS_FUNCTION</b>     | An attempt was made to issue a function that is invalid in a micro presentation space.  |
| <b>PMERR_INV_MICROPS_ORDER</b>        | An attempt was made to play a metafile containing orders that are invalid in a micro presentation space.  |
| <b>PMERR_INV_MIX_ATTR</b>             | An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask. |
| <b>PMERR_INV_MODE_FOR_OPEN_DYN</b>    | An attempt was made to open a segment with the ATTR_DYNAMIC segment set, while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.         |
| <b>PMERR_INV_MODE_FOR_REOPEN_SEG</b>  | An attempt was made to reopen an existing segment while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.                                |
| <b>PMERR_INV_MODIFY_PATH_MODE</b>     | An invalid mode parameter was specified with GpiModifyPath.   |
| <b>PMERR_INV_MULTIPLIER</b>           | An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.   |
| <b>PMERR_INV_NESTED_FIGURES</b>       | Nested figures have been detected within a path definition.   |
| <b>PMERR_INV_OR_INCOMPAT_OPTIONS</b>  | An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.                        |
| <b>PMERR_INV_ORDER_LENGTH</b>         | An invalid order length was detected during GpiPutData or segment drawing.  |
| <b>PMERR_INV_ORDERING_PARM</b>        | An invalid order parameter was specified with GpiSetSegmentPriority.  |

|                                       |   |
|---------------------------------------|---|
| <b>PMERR_INV_OUTSIDE_DRAW_MODE</b>    | An attempt was made to issue a GpiSavePS or GpiRestorePS function, or an output only function (for example, GpiPaintRegion) from GpiPlayMetaFile without the drawing mode set to DM_DRAW. |
| <b>PMERR_INV_PAGE_VIEWPORT</b>        | An invalid viewport parameter was specified with GpiSetPageViewport.  |
| <b>PMERR_INV_PATH_ID</b>              | An invalid path identifier parameter was specified.   |
| <b>PMERR_INV_PATTERN_ATTR</b>         | An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.                                |
| <b>PMERR_INV_PATTERN_REF_PT_ATTR</b>  | An invalid reftpoint attribute value was specified.   |
| <b>PMERR_INV_PATTERN_SET_ATTR</b>     | An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.                                   |
| <b>PMERR_INV_PATTERN_SET_FONT</b>     | An attempt was made to use an unsuitable font as a pattern set.   |
| <b>PMERR_INV_PICK_APERTURE_OPTION</b> | An invalid options parameter was specified with GpiSetPickApertureSize.   |
| <b>PMERR_INV_PICK_APERTURE_POSN</b>   | An invalid pick aperture position was specified.  |
| <b>PMERR_INV_PICK_APERTURE_SIZE</b>   | An invalid size parameter was specified with GpiSetPickApertureSize.  |
| <b>PMERR_INV_PLAY_METAFILE_OPTION</b> | An invalid option parameter was specified with GpiPlayMetaFile.   |
| <b>PMERR_INV_PRIMITIVE_TYPE</b>       | An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.  |
| <b>PMERR_INV_PS_SIZE</b>              | An invalid size parameter was specified with GpiCreatePS or GpiSetPS.   |
| <b>PMERR_INV_PUTDATA_FORMAT</b>       | An invalid format parameter was specified with GpiPutData.  |
| <b>PMERR_INV_QUERY_ELEMENT_NO</b>     | An invalid start parameter was specified with DevQueryCaps.   |
| <b>PMERR_INV_RECT</b>                 | An invalid rectangle parameter was specified.   |

|                                    |  |
|------------------------------------|--|
| <b>PMERR_INV_REGION_CONTROL</b>    | An invalid control parameter was specified with GpiQueryRegionRects.   |
| <b>PMERR_INV_REGION_MIX_MODE</b>   | An invalid mode parameter was specified with GpiCombineRegion.   |
| <b>PMERR_INV_REPLACE_MODE_FUNC</b> | An attempt was made to issue GpiPutData with the editing mode set to SEGEM_REPLACE.  |
| <b>PMERR_INV_RESERVED_FIELD</b>    | An invalid reserved field was specified.   |
| <b>PMERR_INV_RESET_OPTIONS</b>     | An invalid options parameter was specified with GpiResetPS.  |
| <b>PMERR_INV_RGBCOLOR</b>          | An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor.   |
| <b>PMERR_INV_SCAN_START</b>        | An invalid scanstart parameter was specified with a bit-map function.  |
| <b>PMERR_INV_SEG_ATTR</b>          | An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs. |
| <b>PMERR_INV_SEG_ATTR_VALUE</b>    | An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.   |
| <b>PMERR_INV_SEG_NAME</b>          | An invalid segment identifier was specified.   |
| <b>PMERR_INV_SEG_OFFSET</b>        | An invalid offset parameter was specified with GpiPutData.   |
| <b>PMERR_INV_SEGLEN</b>            | An order length exceeds the remaining segment length in the input PIF.   |
| <b>PMERR_INV_SETID</b>             | An invalid setid parameter was specified.  |
| <b>PMERR_INV_SHARPNESS_PARM</b>    | An invalid sharpness parameter was specified with GpiPolyFilletSharp.  |
| <b>PMERR_INV_STOP_DRAW_VALUE</b>   | An invalid value parameter was specified with GpiSetStopDraw.  |
| <b>PMERR_INV_TRANSFORM_TYPE</b>    | An invalid options parameter was specified with a transform matrix function.   |
| <b>PMERR_INV_TYPE</b>              | Invalid file-type parameter.   |
| <b>PMERR_INV_USAGE_PARM</b>        | An invalid options parameter was specified with GpiCreateBitmap.   |
| <b>PMERR_INV_VIEWING_LIMITS</b>    | An invalid limits parameter was specified with GpiSetViewingLimits.  |

|                                       |   |
|---------------------------------------|---|
| <b>PMERR_INV_VIEWLIM</b>              | A set viewing limits order has an inconsistent mask and order length in the input PIF.  |
| <b>PMERR_INV_XFORM</b>                | A set (default) viewing transform order has an inconsistent mask and order length in the input PIF.                                     |
| <b>PMERR_INV_3DCOORD</b>              | An order specifying 3-dimensional coordinates has been found in the input PIF.  |
| <b>PMERR_INVALID_APPL</b>             | Attempted to start an application whose type is not recognized by OS/2.   |
| <b>PMERR_INVALID_ARRAY_COUNT</b>      | An array has an invalid count, that is, less than or equal to zero.   |
| <b>PMERR_INVALID_ARRAY_SIZE</b>       | A control data type array size is invalid.  |
| <b>PMERR_INVALID_ASCIIZ</b>           | The profile string is not a valid zero-terminated string.   |
| <b>PMERR_INVALID_ATOM</b>             | The specified atom does not exist in the atom table.  |
| <b>PMERR_INVALID_ATOM_NAME</b>        | An invalid atom name string was passed.   |
| <b>PMERR_INVALID_BUNDLE_TYPE</b>      | An invalid bundle type was passed.  |
| <b>PMERR_INVALID_CHARACTER_INDEX</b>  | On WinNextChar or WinPrevChar, a character index is invalid, that is, it is less than 1 or is greater than the string length+1.         |
| <b>PMERR_INVALID_CONTROL_DATATYPE</b> | An invalid control data type was specified.   |
| <b>PMERR_INVALID_DATATYPE</b>         | An invalid data type was specified.   |
| <b>PMERR_INVALID_DST_CODEPAGE</b>     | The destination code page parameter is invalid.   |
| <b>PMERR_INVALID_ERRORINFO_HANDLE</b> | On WinFreeErrorInfo, the ERRINFO is not the handle of an ERRINFO structure, that is, it was not created by WinGetErrorInfo.             |
| <b>PMERR_INVALID_FLAG</b>             | An invalid bit was set for a parameter. Use constants defined by PM for options, and do not set any reserved bits.                      |
| <b>PMERR_INVALID_FREE_MESSAGE_ID</b>  | An invalid message identifier was specified. The call has completed by assuming the message parameter and reply data types to be ULONG. |
| <b>PMERR_INVALID_GROUP_HANDLE</b>     | An invalid program-group handle was specified.  |

|                                      |   |
|--------------------------------------|---|
| <b>PMERR_INVALID_HACCEL</b>          | An invalid accelerator-table handle was specified.  |
| <b>PMERR_INVALID_HAPP</b>            | The application handle passed to WinTerminateApp does not correspond to a valid session.  |
| <b>PMERR_INVALID_HATOMTBL</b>        | An invalid atom-table handle was specified.   |
| <b>PMERR_INVALID_HEAP_POINTER</b>    | An invalid pointer was found within the heap.   |
| <b>PMERR_INVALID_HEAP_SIZE</b>       | Invalid data was found within the heap.   |
| <b>PMERR_INVALID_HEAP_SIZE_PARM</b>  | Invalid data was found within the heap.   |
| <b>PMERR_INVALID_HEAP_SIZE_WORD</b>  | Invalid data was found within the heap.   |
| <b>PMERR_INVALID_HENUM</b>           | An invalid enumeration handle was specified.  |
| <b>PMERR_INVALID_HHEAP</b>           | An invalid heap handle was specified.   |
| <b>PMERR_INVALID_HMQ</b>             | An invalid message-queue handle was specified.  |
| <b>PMERR_INVALID_HPTR</b>            | An invalid pointer handle was specified.  |
| <b>PMERR_INVALID_HSTRUCT</b>         | An invalid (null) structure handle was specified.   |
| <b>PMERR_INVALID_HWND</b>            | An invalid window handle was specified.   |
| <b>PMERR_INVALID_INI_FILE_HANDLE</b> | An invalid initialization-file handle was specified.  |
| <b>PMERR_INVALID_INTEGER</b>         | The specified atom is not a valid integer atom.   |
| <b>PMERR_INVALID_INTEGER_ATOM</b>    | The specified atom is not a valid integer atom.   |
| <b>PMERR_INVALID_MESSAGE_ID</b>      | A message identifier is invalid.  |
| <b>PMERR_INVALID_NUMBER_OF_PARMS</b> | The number of parameters is invalid.  |
| <b>PMERR_INVALID_NUMBER_OF_TYPES</b> | The function call has an invalid number (zero) of types.  |
| <b>PMERR_INVALID_PARAMETER</b>       | An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT. |
| <b>PMERR_INVALID_PARAMETER_TYPE</b>  | A parameter type is invalid for a bundle mask.  |

**PMERR\_INVALID\_PARAMETERS**

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32 768 to +32 767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

**PMERR\_INVALID\_PARAMETERS****PMERR\_INVALID\_PARM**

A parameter to the function contained invalid data.

**PMERR\_INVALID\_PROGRAM\_HANDLE**

An invalid program handle was specified.

**PMERR\_INVALID\_SESSION\_ID**

The specified session identifier is invalid. Either zero (for the application's own session) or a valid identifier must be specified.

**PMERR\_INVALID\_SRC\_CODEPAGE**

The source code page parameter is invalid.

**PMERR\_INVALID\_STRING\_PARM**

The specified string parameter is invalid.

**PMERR\_INVALID\_SWITCH\_HANDLE**

An invalid Window List entry handle was specified.

**PMERR\_INVALID\_TARGET\_HANDLE**

An invalid target program-group handle was specified.

**PMERR\_INVALID\_TITLE**

The specified program or group title is too long or contains invalid characters.

**PMERR\_INVALID\_TYPE\_FOR\_LENGTH**

The data type for a control length is invalid.

**PMERR\_INVALID\_TYPE\_FOR\_MPARAM**

The message parameter type for a control MPARAM is invalid, that is, not mparam1, mparam2 or mreply.

**PMERR\_INVALID\_TYPE\_FOR\_OFFSET**

The data type for a control offset is invalid.

**PMERR\_INVALID\_WINDOW**

The window specified with a Window List call is not a valid frame window.

**PMERR\_KERNING\_NOT\_SUPPORTED**

Kerning was requested on GpiCreateLogFont call to a presentation space associated with a device context that does not support kerning.

**PMERR\_LABEL\_NOT\_FOUND**

The specified element label did not exist.

**PMERR\_MATRIX\_OVERFLOW**

An internal overflow error occurred during matrix multiplication. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

**PMERR\_MEMORY\_ALLOC**

An error occurred during memory management.

|  |   |
|--|---|
| <b>PMERR_MEMORY_ALLOCATION_ERR</b>     | An error occurred during memory management.   |
| <b>PMERR_MEMORY_DEALLOCATION_ERR</b>   | An error occurred during memory management.   |
| <b>PMERR_METAFILE_IN_USE</b>           | An attempt has been made to access a metafile that is in use by another thread.   |
| <b>PMERR_METAFILE_INTERNAL_ERROR</b>   | An internal inconsistency has been detected during metafile unlock processing.  |
| <b>PMERR_METAFILE_LIMIT_EXCEEDED</b>   | The maximum permitted metafile size limit was exceeded during metafile recording.   |
| <b>PMERR_MSG_QUEUE_ALREADY_EXISTS</b>  | An attempt to create a message queue for a thread failed because a message queue already exists for the calling thread.   |
| <b>PMERR_MSGID_TOO_SMALL</b>           | The message identifier specified is too small.  |
| <b>PMERR_NEGATIVE_STRCOND_DIM</b>      | A negative array dimension was passed for a data type length.   |
| <b>PMERR_NO_BITMAP_SELECTED</b>        | An attempt has been made to operate on a memory device context that has no bit map selected.  |
| <b>PMERR_NO_CURRENT_ELEMENT</b>        | An attempt has been made to issue <code>GpiQueryElementType</code> or <code>GpiQueryElement</code> while there is no currently open element.  |
| <b>PMERR_NO_CURRENT_SEG</b>            | An attempt has been made to issue <code>GpiQueryElementType</code> or <code>GpiQueryElement</code> while there is no currently open segment.  |
| <b>PMERR_NO_FILL</b>                   | No flood fill occurred because either the starting point color was the same as the input color when a boundary fill was requested, or the starting point color was not the same as the input color when a surface fill was requested.                   |
| <b>PMERR_NO_METAFILE_RECORD_HANDLE</b> | The metafile record handle was not found during metafile recording, or <code>DevEscape (DEVESC_STARTDOC)</code> was not issued when drawing to a <code>OD_QUEUED</code> device context with a <code>pszDataType</code> field of <code>PM_Q_STD</code> . |
| <b>PMERR_NO_MSG_QUEUE</b>              |   |
| <b>PMERR_NO_PALETTE_SELECTED</b>       | An attempt to realize a palette failed because no palette was previously selected into the Presentation Space.  |



|                                       |  |
|---------------------------------------|--|
| <b>PMERR_NO_SPACE</b>                 | The limit on the number of Window List entries has been reached with WinAddSwitchEntry.  |
| <b>PMERR_NOT_CREATED_BY_DEVOPENDC</b> | An attempt has been made to destroy a device context using DevCloseDC that was not created using DevOpenDC.                                |
| <b>PMERR_NOT_CURRENT_PL_VERSION</b>   | An unexpected data format was found in the initialization file.  |
| <b>PMERR_NOT_DRAGGING</b>             | A drag operation is not in progress at this time.  |
| <b>PMERR_NOT_IN_A_PM_SESSION</b>      | An attempt was made to access function that is only available from PM programs from a non-PM session.                                      |
| <b>PMERR_NOT_IN_AREA</b>              | An attempt was made to end an area using GpiEndArea or during segment drawing while not in an area bracket.                                |
| <b>PMERR_NOT_IN_DRAW_MODE</b>         | An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.                                      |
| <b>PMERR_NOT_IN_ELEMENT</b>           | An attempt was made to end an element using GpiEndElement or during segment drawing while not in an element bracket.                       |
| <b>PMERR_NOT_IN_IDX</b>               | The application name, key-name or program handle was not found.  |
| <b>PMERR_NOT_IN_IMAGE</b>             | An attempt was made to end an image during segment drawing while not in an image bracket.  |
| <b>PMERR_NOT_IN_PATH</b>              | An attempt was made to end a path using GpiEndPath or during segment drawing while not in a path bracket.                                  |
| <b>PMERR_NOT_IN_RETAIN_MODE</b>       | An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to <b>retain</b> . |
| <b>PMERR_NOT_IN_SEG</b>               | An attempt was made to end a segment using GpiCloseSegment while not in a segment bracket.   |
| <b>PMERR_NOT_SELF_DESCRIBING_DTYP</b> | A data type is not self-describing.  |
| <b>PMERR_OPENING_INI_FILE</b>         | Unable to open initialization file (due to lack of disk space for example).  |

|  |   |
|--|---|
| <b>PMERR_ORDER_TOO_BIG</b>             | An internal size limit was exceeded while converting orders from short to long format during GpiPutData processing. An order was too long to convert.   |
| <b>PMERR_OWN_SET_ID_REFS</b>           | An attempt to unload a font failed because the setid is still being referenced.   |
| <b>PMERR_PALETTE_BUSY</b>              | An attempt has been made to reset the owner of a palette when it was busy.  |
| <b>PMERR_PALETTE_SELECTED</b>          | Color palette operations cannot be performed on a presentation space while a palette is selected.   |
| <b>PMERR_PARAMETER_OUT_OF_RANGE</b>    | The value of a parameter was not within the defined valid range for that parameter.   |
| <b>PMERR_PATH_INCOMPLETE</b>           | An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.   |
| <b>PMERR_PATH_LIMIT_EXCEEDED</b>       | An internal size limit was exceeded during path or area processing.   |
| <b>PMERR_PATH_UNKNOWN</b>              | An attempt was made to perform a path function on a path that did not exist.  |
| <b>PMERR_PEL_IS_CLIPPED</b>            | An attempt was made to query a pel that had been clipped using GpiQueryPel.   |
| <b>PMERR_PEL_NOT_AVAILABLE</b>         | An attempt was made to query a pel that did not exist in GpiQueryPel (for example, a memory device context with no selected bit map).   |
| <b>PMERR_PRINTER_DD_NOT_DEFINED</b>    | The Presentation Manager device driver has not been defined.  |
| <b>PMERR_PRINTER_QUEUE_NOT_DEFINED</b> | The spooler queue for the printer has not been defined.   |
| <b>PMERR_PRN_ADDR_IN_USE</b>           | A printer is already defined on the port.   |
| <b>PMERR_PRN_ADDR_NOT_DEFINED</b>      | The printer port has not been defined.  |
| <b>PMERR_PRN_NAME_NOT_DEFINED</b>      | The printer has not been defined.   |
| <b>PMERR_PROLOG_ERROR</b>              | A prolog error was detected during drawing. Segment prologs are used internally within retained segments and also appear in metafiles. This error can also arise from an End Prolog order that is outside a prolog. |

|                                      |  |
|--------------------------------------|--|
| <b>PMERR_PS_BUSY</b>                 | An attempt was made to access the presentation space from more than one thread simultaneously.   |
| <b>PMERR_PS_IS_ASSOCIATED</b>        | An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.  |
| <b>PMERR_PS_NOT_ASSOCIATED</b>       | An attempt was made to access a presentation space that is not associated with a device context.   |
| <b>PMERR_QUEUE_TOO_LARGE</b>         | An attempt to create a message queue has failed because the value specified for the size of the message queue is too large.  |
| <b>PMERR_RASTER_FONT</b>             | A request was made for the outline of a bit-map font. Outlines can only be returned for vector font characters.  |
| <b>PMERR_REALIZE_NOT_SUPPORTED</b>   | An attempt was made to create a realizable logical color table on a device driver that does not support this function.   |
| <b>PMERR_REGION_IS_CLIP_REGION</b>   | An attempt was made to perform a region operation on a region that is selected as a clip region.   |
| <b>PMERR_RESOURCE_DEPLETION</b>      | An internal resource depletion error has occurred.   |
| <b>PMERR_RESOURCE_NOT_FOUND</b>      | The specified resource identity could not be found.  |
| <b>PMERR_SEG_AND_REFSEG_ARE_SAME</b> | The segid and refsegid specified with GpiSetSegmentPriority were the same.   |
| <b>PMERR_SEG_CALL_STACK_EMPTY</b>    | A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.   |
| <b>PMERR_SEG_CALL_STACK_FULL</b>     | A call stack full condition was detected when attempting to call a segment using GpiCallSegmentMatrix, attempting to preserve an attribute, or during segment drawing. |
| <b>PMERR_SEG_IS_CURRENT</b>          | An attempt was made to issue GpiGetData to a segment that was currently open.  |
| <b>PMERR_SEG_NOT_CHAINED</b>         | An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.  |

|  |  |
|--|--|
| <b>PMERR_SEG_NOT_FOUND</b>             | The specified segment identifier did not exist.  |
| <b>PMERR_SEG_OVFLOW</b>                | The input PIF has more than 1000 called segments. This has overflowed an internal buffer.                                  |
| <b>PMERR_SEG_STORE_LIMIT_EXCEEDED</b>  | The maximum permitted retained segment store size limit was exceeded.  |
| <b>PMERR_SETID_IN_USE</b>              | An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set. |
| <b>PMERR_SETID_NOT_FOUND</b>           | An attempt was made to delete a setid that did not exist.  |
| <b>PMERR_SMB_OVFLOW</b>                | The input PIF has more than 100 symbol sets defined. This has overflowed an internal buffer.                               |
| <b>PMERR_SOMDD_IS_ACTIVE</b>           | The DSOM daemon is already active.   |
| <b>PMERR_SOMDD_NOT_STARTED</b>         | The DSOM daemon failed to start.   |
| <b>PMERR_SOURCE_SAME_AS_TARGET</b>     | The direct manipulation source and target process are the same.  |
| <b>PMERR_SPL_CANNOT_OPEN_FILE</b>      | Unable to open the file.   |
| <b>PMERR_SPL_DD_NOT_FOUND</b>          | The Presentation Manager device driver definition could not be found.  |
| <b>PMERR_SPL_DEVICE_ALREADY_EXISTS</b> | The device already exists.   |
| <b>PMERR_SPL_DEVICE_LIMIT_REACHED</b>  | The limit on the number of devices has been reached.   |
| <b>PMERR_SPL_DEVICE_NOT_INSTALLED</b>  | The device has not been installed.   |
| <b>PMERR_SPL_DRIVER_ERROR</b>          | No Presentation Manager device driver supplied or found.   |
| <b>PMERR_SPL_DRIVER_NOT_INSTALLED</b>  | The Presentation Manager device driver has not been installed.   |
| <b>PMERR_SPL_FILE_NOT_FOUND</b>        | Unable to find the file.   |
| <b>PMERR_SPL_HARD_NETWORK_ERROR</b>    | Hard network error.  |
| <b>PMERR_SPL_INI_FILE_ERROR</b>        | Error accessing the initialization file.   |
| <b>PMERR_SPL_INV_DATATYPE</b>          | The spool file data type is invalid.   |
| <b>PMERR_SPL_INV_DRIVER_DATATYPE</b>   | The data type is invalid for the Presentation Manager device driver.   |
| <b>PMERR_SPL_INV_FORMS_CODE</b>        | The forms code for the job is invalid.   |
| <b>PMERR_SPL_INV_HSPL</b>              | The spooler handle is invalid.   |

|  |   |
|--|---|
| <b>PMERR_SPL_INV_JOB_ID</b>            | The job id is invalid.  |
| <b>PMERR_SPL_INV_LENGTH_OR_COUNT</b>   | The length or count is invalid.   |
| <b>PMERR_SPL_INV_PRIORITY</b>          | The priority for the job is invalid.  |
| <b>PMERR_SPL_INV_PROCESSOR_DATYPE</b>  | The data type is invalid for the spooler queue processor.                                   |
| <b>PMERR_SPL_INV_QUEUE_NAME</b>        | The spooler queue name is invalid.  |
| <b>PMERR_SPL_INV_TOKEN</b>             | The token is invalid.   |
| <b>PMERR_SPL_JOB_NOT_PRINTING</b>      | The print job is not printing.  |
| <b>PMERR_SPL_JOB_PRINTING</b>          | The print job is already printing.  |
| <b>PMERR_SPL_MANY_QUEUES_ASSOC</b>     | More than one queue has been associated with the printer.                                   |
| <b>PMERR_SPL_NO_CURRENT_FORMS_CODE</b> | There is no current forms code defined to the Presentation Manager device driver.           |
| <b>PMERR_SPL_NO_DATA</b>               | No data supplied or found.  |
| <b>PMERR_SPL_NO_DEFAULT_QUEUE</b>      | There is no default spooler queue for the printer.  |
| <b>PMERR_SPL_NO_DISK_SPACE</b>         | There is not enough free disk space.  |
| <b>PMERR_SPL_NO_FREE_JOB_ID</b>        | There is no free job id available.  |
| <b>PMERR_SPL_NO_MEMORY</b>             | There is not enough free memory.  |
| <b>PMERR_SPL_NO_QUEUES_ASSOCIATED</b>  | A queue has not been associated with the printer.   |
| <b>PMERR_SPL_NO_SUCH_LOG_ADDRESS</b>   | The logical address does not exist (that is, it is not defined in the initialization file). |
| <b>PMERR_SPL_NOT_AUTHORIZED</b>        | Not authorized to perform the operation.  |
| <b>PMERR_SPL_PRINT_ABORT</b>           | The job has already been aborted.   |
| <b>PMERR_SPL_PRINTER_NOT_FOUND</b>     | The printer definition could not be found.  |
| <b>PMERR_SPL_PROCESSOR_ERROR</b>       | No spooler queue processor supplied or found.   |
| <b>PMERR_SPL_PROCESSOR_NOT_INST</b>    | The spooler queue processor has not been installed.   |
| <b>PMERR_SPL_QUEUE_ALREADY_EXISTS</b>  | The spooler queue already exists.   |
| <b>PMERR_SPL_QUEUE_ERROR</b>           | No spooler queue supplied or found.   |
| <b>PMERR_SPL_QUEUE_NOT_EMPTY</b>       | The spooler queue contains print jobs.  |
| <b>PMERR_SPL_QUEUE_NOT_FOUND</b>       | The spooler queue definition could not be found.  |
| <b>PMERR_SPL_SPOOLER_NOT_INSTALLED</b> | The spooler is not installed.   |

|  |  |
|--|--|
| <b>PMERR_SPL_STATUS_STRING_TRUNC</b>   | The print job status string has been truncated.  |
| <b>PMERR_SPL_TEMP_NETWORK_ERROR</b>    | Temporary network error.   |
| <b>PMERR_SPL_TOO_MANY_OPEN_FILES</b>   | Too many open files.   |
| <b>PMERR_SPOOLER_QP_NOT_DEFINED</b>    | The spooler queue processor has not been defined.  |
| <b>PMERR_START_POINT_CLIPPED</b>       | The starting point specified for flood fill is outside the current clipping path or region.                            |
| <b>PMERR_STARTDOC_NOT_ISSUED</b>       | A request to write spooled output without first issuing a STARTDOC was attempted.                                      |
| <b>PMERR_STARTED_IN_BACKGROUND</b>     | The application started a new session in the background.   |
| <b>PMERR_STOP_DRAW_OCCURRED</b>        | Segment drawing or GpiPlayMetaFile was stopped prematurely in response to a GpiSetStopDraw request.                    |
| <b>PMERR_TOO_MANY_METAFILES_IN_USE</b> | The maximum number of metafiles allowed for a given process was exceeded.  |
| <b>PMERR_TRUNCATED_ORDER</b>           | An incomplete order was detected during segment processing.  |
| <b>PMERR_UNABLE_TO_CLOSE_DEVICE</b>    | Unable to close the print device (for example, powered off or offline).  |
| <b>PMERR_UNCHAINED_SEG_ZERO_INV</b>    | An attempt was made to open segment with segment identifier zero and the ATTR_CHAINED segment attribute not specified. |
| <b>PMERR_UNSUPPORTED_ATTR</b>          | An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.                              |
| <b>PMERR_UNSUPPORTED_ATTR_VALUE</b>    | An attribute value was specified with GpiSetAttrs that is not supported.   |
| <b>PMERR_WIN_DEBUGMSG</b>              | Ignore this error. It is reserved for system use.  |
| <b>PMERR_WINDOW_LOCK_OVERFLOW</b>      | An overflow occurred for the use count of a window.  |
| <b>PMERR_WINDOW_LOCK_UNDERFLOW</b>     | An attempt was made to decrement the use count of a window below zero.   |
| <b>PMERR_WINDOW_NOT_LOCKED</b>         | The window specified in WinSendMessage was not locked.   |
| <b>PMERR_WPDSEVER_IS_ACTIVE</b>        | The Workplace Shell DSOM Server is already active.   |

**PMERR\_WPDSERVER\_NOT\_STARTED**

The Workplace Shell DSOM Server could not be started.

**WPERR\_INVALID\_FLAGS**

An invalid flag was specified.

**WPERR\_INVALID\_OBJECTID**

An invalid object ID was specified.

**WPERR\_INVALID\_TARGET\_OBJECT**

An invalid target object was specified.

---

## Appendix D. Standard Bit-Map Formats

There are four standard bit-map formats. All device drivers must be able to translate between any of these formats and their own internal formats. The standard formats are:

| Bitcount | Planes |
|----------|--------|
| 1        | 1      |
| 4        | 1      |
| 8        | 1      |
| 24       | 1      |

These formats are chosen because they are identical or similar to all formats commonly used by raster devices. Only single-plane formats are standard, but it is very easy to convert these to any multiple-plane format used internally by a device.

### Bit-Map Data

The pel data is stored in the bit map in the order that the coordinates appear on a display screen. That is, the pel in the lower-left corner is the first in the bit map. Pels are scanned to the right, and upward, from that position. The bits of the first pel are stored, beginning with the most significant bits of the first byte. The data for pels in each scan line is packed together tightly, but all scan lines are padded at the end, so that each one begins on a ULONG boundary.

### Bit-Map Information Tables

Each standard-format bit map must be accompanied by a bit-map information table. Because the standard-format bit maps are intended to be traded between devices, the color indexes in the bit map are meaningless without more information; for a description of this structure, see BITMAPINFO2.

Some functions use a structure that is similar to BITMAPINFO2 but does not have the color table array; for a description of this structure, see BITMAPINFOHEADER2. Wherever BITMAPINFO2 is shown, BITMAPINFO is also allowed. Similarly, wherever BITMAPINFOHEADER2 is shown, BITMAPINFOHEADER is also allowed.



## Bit-Map Example

To make the ordering of all the bytes clear, consider this simple example of a 5-by-3 array of colored pels:

```
Red  Green Blue Red  Green
Blue Red  Green Blue Red
Green Blue Red  Green Blue
```

```
ULONG ExampleBitmap[] {
    0x23,0x12,0x30,0x00          /* bottom line */
    0x31,0x23,0x10,0x00          /* middle line */
    0x12,0x31,0x20,0x00          /* top line   */
};

#define BLACK 0x00000000L
#define RED   0x00FF0000L
#define GREEN 0x0000FF00L
#define BLUE  0x000000FFL

struct BitmapInfoTable ExampleInfo = {
    5,          /* width   */
    3,          /* height  */
    1,          /* planes  */
    4,          /* bitcount */
    BLACK,RED,GREEN,BLUE, /* color table */
    BLACK,BLACK,BLACK,BLACK,
    BLACK,BLACK,BLACK,BLACK,
    BLACK,BLACK,BLACK,BLACK
};
```

## Bit-Map File Format

The operating system uses the same file format for bit maps, icons, and pointers in resource files. In the following description, "bit map" refers to bit maps, icons, and pointers unless otherwise specified.

Two formats are supported. In the first, a single-size version of the bit map is defined. This is used whatever the target device.

The second format allows multiple versions of the bit map to be defined, including one or more device-independent versions, and a number of device-dependent versions, each intended for use with a particular device.

In the case of icons and pointers, when more than one version of the bit map exists, the preferred version is one that matches the device size of the icon or pointer; otherwise, the device-independent version is used to scale a bit map to the required size.

The operating system provides pointers that match the requirements of the display device in use, typically pointers are 32x32 pels, one bit per plane.

Icons provided with the operating system are designed to match the requirements of the most common display devices. The following versions of each icon are included in each file:

- 32x32 4 bpp (16 color)
- 40x40 4 bpp (16 color)
- 32x32 1 bpp (black and white)
- 20x20 1 bpp (black and white)
- 16x16 1 bpp (black and white)

The 32x32 versions are designed for VGA displays and for device-independent use.

The 40x40 version is for 8514/A and XGA displays.

The 20x20 and 16x16 are half-size icons designed for use as mini-icons.

For general bit maps, which may be of arbitrary size, the preferred version is one matching the requested bit map size; otherwise one matching the display size is selected. If neither is available, the device-independent version is used from which to scale a bit map.

For both formats, the definition consists of two sections. The first section contains general information about the type, dimensions, and other attributes of the resource. The second section contains data describing the pels that make up the bit map(s), and is in the format specified in "Bit-Map Data" on page D-1.

In the multiple-version format, the first section contains an array of BITMAPARRAYFILEHEADER or BITMAPARRAYFILEHEADER2 structures. The format of these structures are as follows:

```
typedef struct _BITMAPARRAYFILEHEADER {
    USHORT          usType;
    ULONG           cbSize;
    ULONG           offNext;
    USHORT          cxDisplay;
    USHORT          cyDisplay;
    BITMAPFILEHEADER bfh;
} BITMAPARRAYFILEHEADER;

typedef BITMAPARRAYFILEHEADER *PBITMAPARRAYFILEHEADER;
```

```
typedef struct _BITMAPARRAYFILEHEADER2 {
    USHORT          usType;
    ULONG           cbSize;
    ULONG           offNext;
    USHORT          cxDisplay;
    USHORT          cyDisplay;
    BITMAPFILEHEADER2 bfh2;
} BITMAPARRAYFILEHEADER2;

typedef BITMAPARRAYFILEHEADER2 *PBITMAPARRAYFILEHEADER2;
```

The device-independent version must be the first BITMAPARRAYFILEHEADER or BITMAPARRAYFILEHEADER2 defined.

In the single-size format, the BITMAPARRAYFILEHEADER or BITMAPARRAYFILEHEADER2 structure is not present. The definition consists of one or two BITMAPFILEHEADER or BITMAPFILEHEADER2 structures.

The format of the BITMAPFILEHEADER and BITMAPFILEHEADER2 structure are defined below:

```
typedef struct _BITMAPFILEHEADER {
USHORT          usType;
ULONG           cbSize;
SHORT           xHotspot;
SHORT           yHotspot;
USHORT          offBits;
BITMAPINFOHEADER bmp;
} BITMAPFILEHEADER;

typedef BITMAPFILEHEADER *PBITMAPFILEHEADER;
```

```
typedef struct _BITMAPFILEHEADER2 {
USHORT          usType;
ULONG           cbSize;
SHORT           xHotspot;
SHORT           yHotspot;
USHORT          offBits;
BITMAPINFOHEADER2 bmp2;
} BITMAPFILEHEADER2;

typedef BITMAPFILEHEADER2 *PBITMAPFILEHEADER2;
```

For icons and pointers, the *cy* field in *bmp* is actually twice the pel height of the image that appears on the screen. This is because these types actually contain two full bit-map pel definitions. The first bit-map definition is the XOR mask, which contains invert information (0 = no invert, 1 = invert) for the pointer or icon. The second is the AND mask, which determines whether the pointer or the screen is shown (0 = black/white, 1 = screen/inverse screen).

For color icons or pointers, there are two bit-maps involved: one that is black and white and consists of an AND and an XOR mask, and one that is color that defines the color content.

The *cy* field in the BITMAPINFOHEADER2 structure for the color bit-map must be the real height, that is, half the value specified for the black and white bit-map. The *cx* fields must be the same.

The following table shows how these two bit-maps are used for a color icon or pointer:

| XOR | AND | COLOR |               |
|-----|-----|-------|---------------|
| 1   | 1   | x     | Invert screen |
| 0   | 0   | x     | Use color x   |
| 0   | 1   | x     | Transparency  |
| 1   | 0   | x     | Use color x   |

For color icons or pointers, two BITMAPFILEHEADER or BITMAPFILEHEADER2 structures are therefore required:

```
BITMAPFILEHEADER2  with usType BFT_COLORICON or BFT_COLORPOINTER
  BITMAPINFOHEADER2 (part of BITMAPFILEHEADER2)
  Color table
BITMAPFILEHEADER2  with same usType
  BITMAPINFOHEADER2 (part of BITMAPFILEHEADER2)
  Color table
**
bits for one bit-map
**
**
bits for other bit-map
**
```

The *usType* for the first BITMAPFILEHEADER2 is either BFT\_COLORICON or BFT\_COLORPOINTER. This means that a second BITMAPFILEHEADER2 is present as part of the definition of a color icon or pointer. The first The first BITMAPFILEHEADER2 structure contains the information for the black and white AND and XOR masks, while the second BITMAPFILEHEADER2 structure contains the information for the color part of the pointer or icon.

BITMAPFILEHEADER and BITMAPINFOHEADER can occur in place of BITMAPFILEHEADER2 and BITMAPINFOHEADER2 in this example.

For the multiple version format, the file is as follows:

```
BITMAPARRAYFILEHEADER2  for device-independent version
  BITMAPFILEHEADER2      (part of BITMAPARRAYFILEHEADER2)
    BITMAPINFOHEADER2    (part of BITMAPFILEHEADER2)
      Color table

  BITMAPFILEHEADER2      )
    BITMAPINFOHEADER2    ) only if this is a color icon or pointer
      Color table        )

BITMAPARRAYFILEHEADER2  for first device-dependent version
  BITMAPFILEHEADER2      (part of BITMAPARRAYFILEHEADER2)
    BITMAPINFOHEADER2    (part of BITMAPFILEHEADER2)
      Color table

  BITMAPFILEHEADER2      )
    BITMAPINFOHEADER2    ) only if this is a color icon or pointer
      Color table        )
```

Further BITMAPARRAYFILEHEADER2 groups occur here as required for additional device-dependent versions

```
**
bits for one bit-map
**
**
bits for next bit-map
**
```

And so on for as many bit-maps as necessary.

As before, BITMAPARRAYFILEHEADER, BITMAPFILEHEADER, and BITMAPINFOHEADER, can occur in place of BITMAPARRAYFILEHEADER2, BITMAPFILEHEADER2, and BITMAPINFOHEADER2,

---

## Appendix E. Fonts Supplied with the OS/2 Operating System

OS/2\* outline fonts and Presentation Manager\* bit map fonts are supplied by the operating system.

---

### OS/2 Outline Fonts

The following Adobe\*\* Type 1 fonts are supplied with OS/2:

| Family Name       | Face Name  |
|-------------------|--|
| Times New Roman** | Times New Roman<br>Times New Roman Bold<br>Times New Roman Bold Italic<br>Times New Roman Italic |
| Helvetica**       | Helvetica<br>Helvetica Bold<br>Helvetica Bold Italic<br>Helvetica Italic                         |
| Courier           | Courier<br>Courier Bold<br>Courier Bold Italic<br>Courier Italic                                 |
| Symbol            | Symbol   |

The Courier, Tms Rmn, and Swiss family fonts that were supplied with OS/2 release 1.1 and 1.2 are no longer supplied. Using one of the old names results in one of the new fonts listed above being used, as follows:

| Old Family/Face Name | Font Used.      |
|----------------------|-----------------|
| Roman/Tms Rmn        | Times New Roman |
| Swiss/Helv           | Helvetica       |

These fonts are provided in an efficient binary format for use by the OS/2 Adobe Type Manager. They are also provided in standard Type 1 format (PFB and AFM) for use with the OS/2 PostScript\*\* printer device driver.

---

\* Trademark of the IBM Corporation.

\*\* Trademarks of Adobe Systems Incorporated, Monotype, and Linotype.

---

## Presentation Manager Bit Map Fonts

The following tables list all system bit map fonts available using the Graphics Programming Interface. The first table applies to hardware that does not conform to the International Standards Organization (ISO) 9241. (See "International Standards Organization (ISO) 9241" on page E-7 for more information on ISO 9241.) The second table lists the fonts supplied with OS/2 for IBM hardware that does conform to ISO 9241.

During system installation, the operating system determines the type of display adapter available on your computer and installs only the fonts which match the device resolution. Since additional device bit map fonts may be available on specific devices, you may have to install the correct bit map fonts if you change your display device after the operating system is installed.

### Fonts Supplied for ISO 9241 Non-Conforming Hardware

The following information for each font is included in the table:

- Points** This is the point size of the font, on a device whose resolution matches that of the font, (see "Device" below).
- Ave Wid** This is the average width in pels of alphabetic characters weighted according to US English letter frequencies.
- Max Wid** This is the maximum width in pels of all characters in the font. This field is not necessarily the maximum width of any character in the code page. It could be used to ensure that the horizontal space allocated on a display or printer is big enough to handle any character.
- Height** This is the height in pels of the font. This is the minimum number of rows of pels needed to output any character of the font on a given baseline. This field may be larger than necessary for a given code page. It could be used to ensure that the vertical space allocated on a display or printer is big enough to handle any character.
- Device** This is the X and Y resolution in pels per inch at which the font is intended to be used. Only those fonts which match the device resolution of the installed display driver are available on the system. If the installed display is changed, the install process will reinstall the proper font sets for the new adapter. The IBM devices whose device drivers report these resolutions are:

|           |                                     |
|-----------|-------------------------------------|
| 96 x 48   | CGA                                 |
| 96 x 72   | EGA                                 |
| 96 x 96   | VGA and XGA (in 640 x 480 mode)     |
| 120 x 120 | 8514/A and XGA (in 1024 x 768 mode) |

**Note:** These values are approximate representations of the actual resolution, which in the case of displays depends on which monitor is attached. Consequently the point size of characters on the screen is also approximate.

The following table applies to hardware that does not conform to ISO 9241.

| Family              | Face Name           | Points | Av Wid | Max Wid | Height | Device  |         |         |
|---------------------|---------------------|--------|--------|---------|--------|---------|---------|---------|
| Courier             | Courier             | 8      | 8      | 8       | 7      | 96x48   |         |         |
|                     |                     |        | 8      | 8       | 10     | 96x72   |         |         |
|                     |                     |        | 8      | 8       | 13     | 96x96   |         |         |
|                     |                     | 10     |        |         | 9      | 9       | 16      | 120x120 |
|                     |                     |        |        |         | 9      | 9       | 8       | 96x48   |
|                     |                     |        |        |         | 9      | 9       | 12      | 96x72   |
|                     |                     |        |        |         | 9      | 9       | 16      | 96x96   |
|                     |                     |        |        |         | 12     | 12      | 20      | 120x120 |
|                     |                     |        |        |         | 12     | 12      | 10      | 96x48   |
|                     |                     |        |        |         | 12     | 12      | 15      | 96x72   |
|                     |                     |        |        |         | 12     | 12      | 20      | 96x96   |
|                     |                     |        |        |         | 15     | 15      | 25      | 120x120 |
| System Proportional | System Proportional | 8      | 6      | 20      | 8      | 96x48   |         |         |
|                     |                     |        | 10     | 6       | 20     | 12      | 96x96   |         |
|                     |                     |        | 10     | 6       | 20     | 16      | 96x96   |         |
|                     |                     |        | 10     | 8       | 23     | 20      | 120x120 |         |
|                     |                     |        | 11     | 10      | 23     | 23      | 120x120 |         |
| System Monospaced   | System Monospaced   | 8      | 8      | 8       | 8      | 96x48   |         |         |
|                     |                     |        | 10     | 8       | 8      | 12      | 96x72   |         |
|                     |                     |        | 10     | 8       | 8      | 16      | 96x96   |         |
|                     |                     |        | 10     | 9       | 9      | 20      | 120x120 |         |
| Helv                | Helv                | 8      | 5      | 13      | 6      | 96x48   |         |         |
|                     |                     |        | 5      | 13      | 10     | 96x72   |         |         |
|                     |                     |        | 5      | 13      | 13     | 96x96   |         |         |
|                     |                     |        | 6      | 14      | 16     | 120x120 |         |         |
|                     |                     |        | 10     | 6       | 15     | 8       | 96x48   |         |
|                     |                     |        | 6      | 14      | 12     | 96x72   |         |         |
|                     |                     |        | 6      | 14      | 16     | 96x96   |         |         |
|                     |                     |        | 7      | 20      | 20     | 120x120 |         |         |
|                     |                     |        | 12     | 7       | 17     | 10      | 96x48   |         |
|                     |                     |        | 7      | 17      | 15     | 96x72   |         |         |
|                     |                     |        | 7      | 17      | 20     | 96x96   |         |         |



| Family  | Face Name | Points | Av Wid | Max Wid | Height | Device  |
|---------|-----------|--------|--------|---------|--------|---------|
|         |           |        | 9      | 21      | 25     | 120x120 |
|         |           | 14     | 8      | 21      | 12     | 96x48   |
|         |           |        | 8      | 21      | 18     | 96x72   |
|         |           |        | 8      | 21      | 24     | 96x96   |
|         |           |        | 11     | 26      | 29     | 120x120 |
|         |           | 18     | 11     | 26      | 15     | 96x48   |
|         |           |        | 10     | 26      | 22     | 96x72   |
|         |           |        | 11     | 26      | 29     | 96x96   |
|         |           |        | 13     | 34      | 36     | 120x120 |
|         |           | 24     | 14     | 35      | 19     | 96x48   |
|         |           |        | 14     | 35      | 28     | 96x72   |
|         |           |        | 14     | 35      | 37     | 96x96   |
|         |           |        | 18     | 45      | 46     | 120x120 |
| Tms Rmn | Tms Rmn   | 8      | 4      | 12      | 6      | 96x48   |
|         |           |        | 4      | 13      | 10     | 96x72   |
|         |           |        | 4      | 12      | 13     | 96x96   |
|         |           |        | 5      | 14      | 16     | 120x120 |
|         |           | 10     | 6      | 15      | 8      | 96x48   |
|         |           |        | 5      | 14      | 12     | 96x72   |
|         |           |        | 5      | 14      | 16     | 96x96   |
|         |           |        | 7      | 19      | 20     | 120x120 |
|         |           | 12     | 7      | 18      | 10     | 96x48   |
|         |           |        | 6      | 18      | 15     | 96x72   |
|         |           |        | 6      | 16      | 19     | 96x96   |
|         |           |        | 8      | 23      | 23     | 120x120 |
|         |           | 14     | 7      | 21      | 11     | 96x48   |
|         |           |        | 7      | 21      | 16     | 96x72   |
|         |           |        | 7      | 20      | 21     | 96x96   |
|         |           |        | 10     | 26      | 27     | 120x120 |
|         |           | 18     | 10     | 26      | 14     | 96x48   |
|         |           |        | 10     | 26      | 20     | 96x72   |
|         |           |        | 10     | 26      | 27     | 96x96   |
|         |           |        | 12     | 34      | 33     | 120x120 |
|         |           | 24     | 14     | 35      | 18     | 96x48   |
|         |           |        | 13     | 35      | 26     | 96x72   |

| Family | Face Name | Points | Av Wid | Max Wid | Height | Device  |
|--------|-----------|--------|--------|---------|--------|---------|
|        |           |        | 13     | 35      | 35     | 96x96   |
|        |           |        | 16     | 46      | 43     | 120x120 |

## Fonts Supplied for ISO 9241 Conforming Hardware

The following table lists the fonts and sizes that have been tested and certified as passing the ISO 9241 black text on white background criteria for the three IBM displays that conform to the standard. These displays are:

- 9515 - A 14 inch XGA display.
- 9517 - A 17 inch XGA display.
- 9518 - A 14 inch VGA display.

See "International Standards Organization (ISO) 9241" on page E-7 for information on ISO 9241.

The following information about each font is also included in the table:

- P** The point size of the font.  
**AW** The average character width in pels in the font.  
**MW** The maximum character width in pels in the font.  
**HE** The height in pels of the font (maximum baseline extent).  
**Device** The X and Y resolution in pels per inch on the device the font is intended to be used. The IBM devices whose device drivers report these resolutions are:  
96 x 96 VGA and XGA (in 640 x 480 mode)  
120 x 120 XGA (in 1024 x 768 mode)

| Family Name | Face Name | P  | AW | MW | HE | Device  | 9515 | 9517 | 9518 |
|-------------|-----------|----|----|----|----|---------|------|------|------|
| Courier     | Courier   | 8  | 8  | 8  | 13 | 96 96   | No   | No   | No   |
|             | ISO       | 8  | 10 | 10 | 16 | 120 120 | No   | No   | n/a  |
|             |           | 9  | 8  | 8  | 15 | 96 96   | Yes  | Yes  | Yes  |
|             |           | 10 | 10 | 10 | 16 | 96 96   | Yes  | Yes  | Yes  |
|             |           | 10 | 12 | 12 | 20 | 120 120 | No   | No   | n/a  |
|             |           | 12 | 12 | 12 | 20 | 96 96   | Yes  | Yes  | Yes  |
|             |           | 12 | 15 | 15 | 25 | 120 120 | Yes  | Yes  | n/a  |

| Family Name         | Face Name           | P  | AW | MW      | HE  | Device  | 9515 | 9517 | 9518 |
|---------------------|---------------------|----|----|---------|-----|---------|------|------|------|
| Helv                | Helv ISO            | 8  | 5  | 13      | 13  | 96 96   | No   | No   | No   |
|                     |                     | 8  | 7  | 14      | 16  | 120 120 | No   | No   | n/a  |
|                     |                     | 9  | 6  | 13      | 15  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 9  | 8  | 20      | 21  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 10 | 7  | 14      | 16  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 9  | 20      | 21  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 12 | 9  | 17      | 20  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 12 | 10 | 21      | 25  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 14 | 10 | 21      | 24  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 14 | 12 | 26      | 29  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 18 | 12 | 26      | 29  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 18 | 15 | 34      | 36  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 24 | 14 | 34      | 36  | 96 96   | Yes  | Yes  | Yes  |
| 24                  | 19                  | 45 | 46 | 120 120 | Yes | Yes     | n/a  |      |      |
| Tms Rmn.            | Tms Rmn ISO         | 8  | 5  | 12      | 13  | 96 96   | No   | No   | No   |
|                     |                     | 8  | 7  | 15      | 16  | 120 120 | No   | No   | n/a  |
|                     |                     | 9  | 6  | 12      | 15  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 7  | 14      | 16  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 8  | 17      | 19  | 120 120 | No   | Yes  | n/a  |
|                     |                     | 12 | 8  | 16      | 19  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 12 | 10 | 23      | 22  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 14 | 9  | 23      | 22  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 14 | 11 | 26      | 27  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 18 | 11 | 26      | 27  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 18 | 14 | 34      | 34  | 120 120 | Yes  | Yes  | n/a  |
|                     |                     | 24 | 14 | 34      | 34  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 24 | 17 | 46      | 43  | 120 120 | Yes  | Yes  | n/a  |
| System Proportional | System Proportional | 9  | 6  | 13      | 15  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 6  | 20      | 16  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 8  | 23      | 20  | 120 120 | No   | Yes  | n/a  |
|                     |                     | 12 | 10 | 23      | 22  | 120 120 | Yes  | Yes  | n/a  |
| System Mono-spaced  | System Mono-spaced  | 10 | 8  | 8       | 16  | 96 96   | Yes  | Yes  | Yes  |
|                     |                     | 10 | 10 | 10      | 21  | 120 120 | Yes  | Yes  | n/a  |

See "International Standards Organization (ISO) 9241" on page E-7 for more information on ISO 9241.

---

## International Standards Organization (ISO) 9241

ISO 9241 is an international standard covering health and safety in the work place for users of visual display terminals. Part 3 of this standard covers clarity and legibility of text displayed on computer screens; it places requirements on minimum sizes and luminance contrast. The presence of the FM\_SEL\_ISO9241\_TESTED flag in the FONTMETRICS structure indicates that the font has been tested for ISO compliance.

**Note:** While the fonts were primarily tested for meeting the ISO standard, they have also been designed to meet the German standard DIN 66 234. Where the two standards differ, the fonts have been designed to meet the stricter requirement.

The FM\_ISO\_xxx flags indicate the results of the test on the three IBM\* displays that conform to the standard. These are the IBM 9515, 9517, and 9518 color displays at the supported resolutions of 640 x 480 and 1024 x 768. To determine whether a non-IBM display complies with ISO 9241, contact the manufacturer. The current display type can be established using VioGetConfig.

In order for applications to meet the standard, they have to ensure that they use only fonts that have been tested and passed. You can determine this by examining the new FM\_SEL\_ISO9241\_TESTED flag in the *fsSelection* parameter in the FONTMETRICS structure, the FM\_ISO\_xxx flags and the *sXDeviceRes* and *sYDeviceRes* fields in the structure.

See Appendix E, "Fonts Supplied with the OS/2 Operating System" on page E-1 for the table describing ISO 9241 compliant fonts.

---

\* Trademark of IBM Corporation.



---

## Appendix F. Format of Interchange Files

A metafile is a file in which graphics are stored. The file is application-created, and it contains the graphics orders generated from those GPI calls that are valid in a metafile. Metafiled graphics can be reused by the application that created them. They can also be made available to other applications at the same, or at a different, workstation.

This section describes the restrictions which apply when generating the metafile and gives detail of the overall structure. For the graphics orders descriptions, see “Graphics Orders” in the *Graphics Programming Interface Programming Reference*.

---

### Metafile Restrictions

The following restrictions apply to the generation of all metafiles, and also to the generation of a PM\_Q\_STD print file to a OD\_QUEUED device:

- If GpiWCBitBit or GpiBitBit is used to copy a bit map to a device context in an application, the application should not delete that bit map handle with GpiDeleteBitmap before the device context is closed (metafile is closed).
- GpiSetPS must not be used.
- GpiSetPageViewport is ignored.

The following section lists some general rules that must be followed when creating a metafile that is to be acceptable to SAA-conforming implementations, or replayed into a presentation space that is in **draw-and-retain** or **retain** mode (see “GpiSetDrawingMode” in *Graphics Programming Interface Programming Reference*).

- These items must be established or defaulted before any drawing occurs to the graphics presentation space, and not changed subsequently:
  - The graphics field (GpiSetGraphicsField). For an SAA-conforming metafile, the graphics field must be defaulted or set to no clipping.
  - The code page for the default character set (GpiSetCp).
  - The color table or palette (GpiCreateLogColorTable or GpiCreatePalette). The size of the color table must not exceed 31KB (KB equals 1024 bytes).
  - The default viewing transform (GpiSetDefaultViewMatrix).
  - The setting of the draw controls (GpiSetDrawControl). DCTL\_DISPLAY must be defaulted or set ON.
  - The default values of attributes (see “GpiSetDefAttr” in the *Graphics Programming Interface Programming Reference*), viewing limits (see “GpiSetDefViewingLimits” in the *Graphics Programming Interface Programming Reference*), primitive tag (see “GpiSetDefTag” in the *Graphics Programming Interface Programming Reference*) and arc parameters (see “GpiSetDefArcParams” in the *Graphics Programming Interface Programming Reference*).

- These calls should not be used:
  - GpiBitBlt
  - GpiDeleteSetId (note that this means that local identifiers cannot be used again within the picture)
  - GpiErase
  - GpiExcludeClipRectangle
  - GpiIntersectClipRectangle
  - GpiOffsetClipRegion
  - GpiPaintRegion
  - GpiResetPS
  - GpiSetClipRegion
  - GpiSetPel
  - GpiSetPS
  - DevEscape (for an escape which is metafiled).
- GpiCreateLogFont must not redefine a local identifier that has previously been used within the picture.
- The metafile context must not be reassociated.
- If a bit map is used as the source of a GpiWCBitBlt operation, or as an area-fill pattern, it must not be modified or deleted (GpiDeleteBitmap) before the metafile is closed.
- Only these foreground mixes must be used (see “GpiSetMix” in the *Graphics Programming Interface Programming Reference*):
  - FM\_DEFAULT
  - FM\_OR
  - FM\_OVERPAINT
  - FM\_LEAVEALONE
- Only these background mixes must be used (see “GpiSetBackMix” in the *Graphics Programming Interface Programming Reference*):
  - BM\_DEFAULT
  - BM\_OVERPAINT
  - BM\_LEAVEALONE
- If palettes are used (see “GpiCreatePalette” in the *Graphics Programming Interface Programming Reference*): the palette that is metafiled is the one in force when the metafile device context is dissociated from the (final) presentation space. If the palette is changed during the course of the picture (using GpiSetPaletteEntries), it must therefore only be with incremental additions.

**Note:** There is no restriction concerning the use of primitives outside segments. These are metafiled in segment(s) with zero identifier.

---

## Metafile Data Format

This section describes the format of the data in a metafile, as it would be stored in an OS/2\* disk file.

Metafile data is stored as a sequence of structured fields. Each structured field starts with an eight-byte header consisting of a two-byte *length* field and a three-byte *identifier* field. These are followed by a one-byte *flags* field and a two-byte *segment sequence number* field.

The length field contains a count of the total number of bytes in the structured field, including the length field. The identifier field uniquely identifies the type of the structured field.

The flags and segment sequence number fields are always zero.

Following the header are positional parameters that are optional and dependent on the particular structured field.

Following the positional parameters are non-positional parameters called *triplets*. These are self-defining parameters and consist of a one-byte *length* field, followed by a one-byte *identifier* field, followed by the data of the parameter.

The length field contains a count of the total number of bytes in the triplet, including the length and identifier fields. The identifier field identifies uniquely the type of the triplet.

A metafile is structured into a number of different functional components; for example, document and graphics object. Each component comprises a number of structured fields, and is delimited by "begin-component" and "end-component" structured fields. Structured fields marked as *required*, inside an *optional* structured field bracket, are required if the containing bracket is present.

The graphics orders that describe a picture occur in the *graphics data* structured field. See "Structured Field Formats" on page F-4 for more information.

---

\* Trademark of IBM Corporation



---

## Structured Field Formats

The format of the various structured fields is given below:

### Begin Document

#### **Structured Field Introducer (BDT): required**

0-1        Length 0xn+1E  
2-4        BDT 0xD3A8A8  
5          Flags 0x00  
6-7        Segment sequence number 0x0000

#### **Parameters**

0-7        Document name C'0000 0001'  
8          Architecture version 0x00  
9          Document security 0x00

#### **Triplets (all required)**

0          Length 0x05  
1          Triplet Id 0x18  
2          Interchange set type 0x03 (resource document)  
3-4        Base set definition 0x0C00 (level 12, version 0)

0          Length 0x06  
1          Triplet Id 0x01  
2-5        GCID

0          Length 0xn+1  
1          Triplet Id 0x65  
2-n        Comment, used for metafile description of up to 252 bytes.

### Begin Resource Group (BRG): required

#### **Structured Field Introducer**

0-1        Length 0x0010  
2-4        BRG 0xD3A8C6  
5          Flags 0x00  
6-7        Segment sequence number 0x0000

#### **Parameters**

0-7        Resource group name C'0000 0002'

**Begin Color Attribute (BCA) Table: required**

**Structured Field Introducer**

- 0-1 Length 0x0010
- 2-4 BCA 0xD3A877
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

**Parameters**

- 0-7 Color table name C'0000 0004'

**Color Attribute Table (CAT): required**

**Structured Field Introducer**

- 0-1 Length 0xn+8
- 2-4 CAT 0xD3B077
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

**Parameters**

**Base Part (required)**

- 0 Flags
  - 0 Reserved B'0'
  - 1 Reset
    - B'0' Do not reset to default
    - B'1' Do reset to default
  - 2-7 Reserved B'000000'
- 1 Reserved 0x00
- 2 LCTID 0x00

Element list(s) (triple generating) are mutually-exclusive. One or other is required.

**Element List (repeating)**

- 0 Length of this parameter
- 1 Type 0x01: element list
- 2 Flags 0x00: reserved
- 3 Format
  - 0x01 RGB
- 4-6 Starting Index (Top Byte Truncated)
- 7 Size of RGB component1 0x08
- 8 Size of RGB component2 0x08
- 9 Size of RGB component3 0x08
- 10 Number of bytes in each following color triple 0x04
- 11-m Color triples

### Triple Generating

|   |                          |                                     |
|---|--------------------------|-------------------------------------|
| 0 | Length of this parameter | 0x0A                                |
| 1 | Type                     | 0x02: bit generator                 |
| 2 | Flags                    |                                     |
|   | 0                        | ABFlag                              |
|   | B'0'                     | Normal                              |
|   | 1-7                      | Reserved B'0000000'                 |
| 3 | Format                   |                                     |
|   | 0x01                     | RGB                                 |
|   | 4-6                      | Starting index (top byte truncated) |
| 7 | Size of RGB component1   | 0x08                                |
| 8 | Size of RGB component2   | 0x08                                |
| 9 | Size of RGB component3   | 0x08                                |

### End Color Attribute (ECA) Table: required

#### Structured Field Introducer

|     |                         |          |
|-----|-------------------------|----------|
| 0-1 | Length                  | 0x0010   |
| 2-4 | ECA                     | 0xD3A977 |
| 5   | Flags                   | 0x00     |
| 6-7 | Segment sequence number | 0x0000   |

#### Parameters

|     |                  |              |
|-----|------------------|--------------|
| 0-7 | Color table name | C'0000 0004' |
|-----|------------------|--------------|

### Begin Image Object (BIM): optional, repeating

#### Structured Field Introducer

|     |                         |          |
|-----|-------------------------|----------|
| 0-1 | Length                  | 0x0010   |
| 2-4 | BIM                     | 0xD3A8FB |
| 5   | Flags                   | 0x00     |
| 6-7 | Segment sequence number | 0x0000   |

#### Parameters

|     |            |              |
|-----|------------|--------------|
| 0-7 | Image name | C'xxxx xxxx' |
|-----|------------|--------------|

### Begin Resource Group (BRG): optional

#### Structured Field Introducer

|     |                         |          |
|-----|-------------------------|----------|
| 0-1 | Length                  | 0x0010   |
| 2-4 | BRG                     | 0xD3A8C6 |
| 5   | Flags                   | 0x00     |
| 6-7 | Segment sequence number | 0x0000   |

#### Parameters

|     |                     |              |
|-----|---------------------|--------------|
| 0-7 | Resource group name | C'xxxx xxxx' |
|-----|---------------------|--------------|

### **Color Attribute Table (BCA): optional**

#### **Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0x0010                  |
| 2-4 | BCA 0xD3A877                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

#### **Parameters**

|     |                               |
|-----|-------------------------------|
| 0-7 | Color table name C'xxxx xxxx' |
|-----|-------------------------------|

### **Color Attribute Table (CAT): required**

#### **Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length                         |
| 2-4 | CAT 0xD3B077                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

#### **Parameters**

##### **Base Part**

|   |               |
|---|---------------|
| 0 | Flags 0x00    |
| 1 | Reserved 0x00 |
| 2 | LUTID         |

##### **Element List (repeating)**

|      |   |
|------|---|
| 0    | Length of this parameter                            |
| 1    | Type 0x01: element list                             |
| 2    | Flags 0x00: reserved                                |
| 3    | Format 0x01: RGB                                    |
| 4-6  | Starting index<br>(top byte truncated)              |
| 7    | Size of RGB component1 0x08                         |
| 8    | Size of RGB component2 0x08                         |
| 9    | Size of RGB component3 0x08                         |
| 10   | Number of bytes in each following color triple 0x03 |
| 11-n | Color triples                                       |

### **End Color Attribute Table (ECA): required if BCA present**

#### **Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0x0010                  |
| 2-4 | ECA 0xD3A977                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

### Parameters

0-7 Color Table name C'xxxx xxxx'

### End Resource Group (ERG): required if BRG present

#### Structured Field Introducer

0-1 Length 0x0010  
2-4 ERG 0xD3A9C6  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-7 Resource Group name C'xxxx xxxx'

### Begin Object Environment Group (BOG): optional

#### Structured Field Introducer

0-1 Length 0x0010  
2-4 BOG 0xD3A8C7  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-7 Object environment group name C'xxxx xxxx'

### Map Color Attribute (MCA) Table: required

#### Structured Field Introducer

0-1 Length 0x001A  
2-4 MCA 0xD3AB77  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-1 Length

### Triplet (required)

0 Length 0x0C  
1 Triplet type: fully qualified name 0x02  
2 Type: ref to Begin Resource Object 0x84  
3 ID 0x00  
4-11 Color table name C'xxxx xxxx'

**Icid (required)**

- 0 Length 0x04
- 1 Triplet type: resource local ID 0x24
- 2 Type color table resource 0x07
- 3 Local identifier (LUT-ID) 0x01

**End Object Environment Group (EOG): required if BOG present****Structured Field Introducer**

- 0-1 Length 0x0010
- 2-4 EOG 0xD3A9C7
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

**Parameters**

- 0-7 Object Environment Group name C'xxxx xxxx'

**Image Data Descriptor (IDD): required****Structured Field Introducer**

- 0-1 Length 0x0011
- 2-4 IDD 0xD3A6FB
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

**Parameters**

- 0 Unit of measure:
  - 0x00 tens of inches
  - 0x01 tens of centimeters
- 1-2 X resolution image points / UOM
- 3-4 Y resolution image points / UOM
- 5-6 X extent of image PS
- 7-8 Y extent of image PS

**Image Picture Data (IPD): required****Structured Field Introducer**

- 0-1 Length
- 2-4 IPD 0xD3EEFB
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

**Parameters (all required and in this order, except that only one of Image LUT-ID and IDE structure is present)**

### Begin Segment

- 0 Type 0x70: begin segment
- 1 Length of following 0x00

### Begin Image Content

- 0 Type 0x91: Begin Image Content
- 1 Length of following 0x01
- 2 Format 0xFF

### Image Size

- 0 Type 0x94: image size
- 1 Length of following 0x09
- 2 Units of measure 0x02: logical
- 3-4 Horizontal resolution
- 5-6 Vertical resolution
- 7-8 Height in pels
- 9-10 Width in pels

### Image Encoding

- 0 Type 0x95: image encoding
- 1 Length of following 0x02
- 2 Compression algorithm 0x03: none
- 3 Recording algorithm 0x03: bottom-to-top

### Image IDE-Size

- 0 Type 0x96: image IDE-Size
- 1 Length of following 0x01
- 2 Number of bits per element

### Image LUT-ID (For bit maps with other than 24 bits per pel)

- 0 Type 0x97 Image LUT-ID
- 1 Length of following 0x01
- 2 LUT-ID

### IDE Structure (For bit maps with 24 bits per pel)

- 0 Type 0x9B: IDE structure
- 1 Length of following 0x08
- 2 Flags:
  - 0 ABFlag
  - B'0' Normal (Additive)
  - 1-7 Reserved B'0000000'
- 3 Format
  - 0x01 RGB

|     |                   |
|-----|-------------------|
| 4-6 | Reserved 0x000000 |
| 7   | Size of element 1 |
| 8   | Size of element 2 |
| 9   | Size of element 3 |

**Image Picture Data (IPD): required, repeating**

**Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length                         |
| 2-4 | IPD 0xD3EEFB                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

**Parameters**

**Image Data**

|     |                                     |
|-----|-------------------------------------|
| 0-1 | Type 0xFE92: image data             |
| 2-3 | Length of following                 |
| 4-n | Image data (scan lines of bit maps) |

**End Image Content (required, only present in last Image Picture Data)**

|   |                              |
|---|------------------------------|
| 0 | Type 0x93: End Image Content |
| 1 | Length of following 0x00     |

**End Segment (required, only present in last Image Picture Data)**

|   |                          |
|---|--------------------------|
| 0 | Type 0x71: end segment   |
| 1 | Length of following 0x00 |

**End Image Object (EIM): required if BIM present**

**Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0x0010                  |
| 2-4 | EIM 0xD3A9FB                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

**Parameters**

|     |                         |
|-----|-------------------------|
| 0-7 | Image name C'xxxx xxxx' |
|-----|-------------------------|

**Begin Graphics Object (BGR): required**

**Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0x0010                  |
| 2-4 | BGR 0xD3A8BB                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |



### Parameters

0-7 Graphics object name C'0000 0007'

### Begin Object Environment Group (BOG): optional

#### Structured Field Introducer

0-1 Length 0x0010  
2-4 LOG 0xD3A8C7  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-7 Object Environment Group name C'0000 0007'

### Map Color Attribute Table (MCA): required

#### Structured Field Introducer

0-1 Length 0x0016  
2-4 MCA 0xD3AB77  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-1 Length

#### Triplet (required)

0 Length 0x0C  
1 Triplet type: fully qualified name 0x02  
2 Type: ref to Begin Resource Object 0x84  
3 ID 0x00  
4-11 Color table name C'0000 0004'

### Map Coded Font (MCF): required, for default font

#### Structured Field Introducer

0-1 Length 0x20  
2-4 MCF 0xD3AB8A  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### Parameters

0-1 Length

#### Triplets (required)

### Font name

- 0 Length 0x0C
- 1 Triplet type: fully qualified name 0x02
- 2 Type: ref to coded font 0x84
- 3 ID 0x00
- 4-11 Coded font name: C'nxxx xxxx'  
where n is 0xFF

### Icid

- 0 Length 0x04
- 1 Triplet type: Resource Local ID 0x24
- 2 Type: Coded Font Resource 0x05
- 3 Local identifier (LCID) 0x00

### Font Binary GCID

- 0 Length 0x06
- 1 Triplet type: Font Binary GCID 0x20
- 2-5 GCID

### Map Coded Font (MCF): optional, repeating, for loaded fonts

#### Structured Field Introducer

- 0-1 Length 0x58
- 2-4 MCF 0xD3AB8A
- 5 Flags 0x00
- 6-7 Segment sequence number 0x0000

#### Parameters

- 0-1 Length

#### Triplets (required)

##### Font name

- 0 Length 0x0C
- 1 Triplet type: fully qualified name 0x02
- 2 Type: ref to coded font 0x84
- 3 ID 0x00
- 4-11 Coded font name

##### Icid

- 0 Length 0x04
- 1 Triplet type: Resource Local ID 0x24
- 2 Type: coded font resource 0x05
- 3 Local identifier (LCID)

### Font Attributes

|       |                                    |
|-------|------------------------------------|
| 0     | Length 0x14                        |
| 1     | Triplet type: Font Descriptor 0x1F |
| 2     | Weight Class                       |
| 3     | Width Class                        |
| 4-5   | Font Height                        |
| 6-7   | Char Width                         |
| 8     | Descript Flags                     |
| 9     | Usage Codes                        |
| 10    | Family                             |
| 11    | Activity Class                     |
| 12    | Font Quality                       |
| 13-14 | CAP Height                         |
| 15-16 | X Height                           |
| 17-18 | Line Density                       |
| 19    | Use Flags                          |

### Font Binary GCID

|     |                                     |
|-----|-------------------------------------|
| 0   | Length 0x06                         |
| 1   | Triplet type: Font Binary GCID 0x20 |
| 2-5 | GCID                                |

### Font Typeface

|      |   |
|------|---|
| 0    | Length 0x24                             |
| 1    | Triplet type: fully qualified name 0x02 |
| 2    | Type: ref to font typeface 0x08         |
| 3    | ID 0x00                                 |
| 4-35 | Font typeface C'xxx...xxx'              |

### Map Data Resource (MDR): optional, repeating

#### Structured Field Introducer

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0x1D                    |
| 2-4 | MDR 0xD3ABC3                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

#### Parameters

|     |        |
|-----|--------|
| 0-1 | Length |
|-----|--------|

#### Triplets (required)

#### Bit-map Name

|   |   |
|---|---|
| 0 | Length 0x0C                             |
| 1 | Triplet type: fully qualified name 0x02 |
| 2 | Type: ref to Image Object 0x84          |

3 ID 0x00  
4-11 Image name C'xxxx xxxx'

**Extended Resource Icid**

0 Length 0x07  
1 Triplet type: Extended Resource Local ID 0x22  
2 Type: Image Resource 0x10  
3-6 Bit-map handle

**End Object Environment Group (EOG): required if BOG present**

**Structured Field Introducer**

0-1 Length 0x0010  
2-4 EOG 0xD3A9C7  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

**Parameters**

0-7 Object Environment Group name C'0000 0007'

**Graphics Data Descriptor (GDD): required**

**Structured Field Introducer**

0-1 Length 0xnxxx  
2-4 GDD 0xD3A6BB  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

**Parameters (all required and in this order)**

0 0xF7 Specify GVM Subset  
1 Length of following data 0x07  
2 0xB0 drawing order subset  
3-4 0x0000  
5 0x23 Level 3.2  
6 0x01 Version 1  
7 Length of following field 0x01  
8 Coordinate types in data  
0x04 Intel16  
0x05 Intel32

0 0xF6 Set Picture Descriptor  
1 Length of following data  
2 Flags  
0 B'0' Picture in 2D  
1 Picture Dimensions  
B'0' Not absolute (PU\_ARBITRARY PS)  
B'1' Absolute (example: PU\_TWIPS PS)

- 2 Picture Elements
    - B'0' Not pels
    - B'1' Pels (PU\_PELS PS)  
(Bit 1 must also be set)
  - 3-7 B'00000'
  - 3 0x00 Reserved
  - 4 Picture frame size coordinate type
    - 0x04 Intel16
    - 0x05 Intel32
  - 5 UnitsOfMeasure
    - 0x00 Ten inches
    - 0x01 Decimeter
  - 6-11 or 6-17 (2 or 4 bytes) Resolution.
    - GPS Units / UOM on x axis
    - GPS Units / UOM on y axis
    - GPS Units / UOM on z axis
  - 12-23 or 18-41 (2 or 4 bytes) Window Size.
    - GPS X left, X right
    - GPS Y bottom, Y top
    - GPS Z near, Z far
- 
- 0 0x21 Set Current Defaults
  - 1 Length of following data
  - 2 Set Default Parameter Format 0x08
  - 3-4 Mask 0xE000
  - 5 Names 0x8F
  - 6 Coordinates
    - 0x00 Picture in 2D
  - 7 Transforms
    - 0x04 Intel16
    - 0x05 Intel32
  - 8 Geometrics
    - 0x04 Intel16
    - 0x05 Intel32
- 
- 0 0x21 Set Current Defaults
  - 1 Length of following data
  - 2 Set default viewing transform 0x07
  - 3-4 Mask 0xCC0C
  - 5 Names 0x8F
  - 6-n M11, M12, M21, M22, M41, M42 Matrix elements
- 
- 0 0x21 Set Current Defaults
  - 1 Length of following data
  - 2 Set default line attributes 0x01
  - 3-4 Mask - OR of as many of the following bits as are required:
    - 0x8000 Line type
    - 0x4000 Line width

**0x2000** Line end  
**0x1000** Line join  
**0x0800** Stroke width  
**0x0008** Line color  
**0x0002** Line mix  
5 Flags  
**0x0F** Set indicated default attributes to initial values. (Data field is not present in this instance).  
**0x8F** Set indicated default attributes to specified values.  
6-n Data – data values as required, in the following order if present.  
No space is reserved for attributes for which the corresponding mask flag was not set.  
**(1 byte)** - Line type  
**(1 byte)** - Line width  
**(1 byte)** - Line end  
**(1 byte)** - Line join  
**(G bytes)** - Stroke width  
**(4 bytes)** - Line color  
**(1 byte)** - Line mix  
(G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)

0 **0x21** Set Current Defaults  
1 Length of following data  
2 Set Default Character Attributes **0x02**  
3-4 Mask – OR of as many of the following bits as are required:  
**0x8000** Character angle  
**0x4000** Character box  
**0x2000** Character direction  
**0x1000** Character precision  
**0x0800** Character set  
**0x0400** Character shear  
**0x0040** Character break extra  
**0x0020** Character extra  
**0x0008** Character color  
**0x0004** Character background color  
**0x0002** Character mix  
**0x0001** Character background mix  
5 Flags  
**0x0F** Set indicated default attributes to initial values. (Data field is not present in this case).  
**0x8F** Set indicated default attributes to specified values.  
6-n Data - data values as required, in the following order if present.  
No space is reserved for attributes for which the corresponding Mask flag was not set.  
**(2\*G bytes)** - Character angle  
**(2\*G + 4 bytes)** - Character box  
**(1 byte)** - Character direction

- (1 byte) - Character precision
  - (1 byte) - Character set
  - (2\*G bytes) - Character shear
  - (4 bytes) - Character break extra
  - (4 bytes) - Character extra
  - (4 bytes) - Character color
  - (4 bytes) - Character background color
  - (1 byte) - Character mix
  - (1 byte) - Character background mix
- (G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)

- 0** 0x21 Set Current Defaults
- 1** Length of following data
- 2** Set Default Marker Attributes 0x03
- 3-4** Mask - OR of as many of the following bits as are required:
- 0x4000** Marker box
  - 0x1000** Marker precision
  - 0x0800** Marker set
  - 0x0100** Marker symbol
  - 0x0008** Marker color
  - 0x0004** Marker background color
  - 0x0002** Marker mix
  - 0x0001** Marker background mix
- 5** Flags
- 0x0F** Set indicated default attributes to initial values.  
(Data field is not present in this instance)
  - 0x8F** Set indicated default attributes to specified values.
- 6-n** Data - data values as required, in this order if present.  
No space is reserved for attributes for which the corresponding Mask flag was not set.
- (2\*G bytes) - Marker box
  - (1 byte) - Marker precision
  - (1 byte) - Marker set
  - (1 byte) - Marker symbol
  - (4 bytes) - Marker color
  - (4 bytes) - Marker background color
  - (1 byte) - Marker mix
  - (1 byte) - Marker background mix
- (G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)
- 0** 0x21 Set Current Defaults
- 1** Length of following data
- 2** Set Default Pattern Attributes 0x04

3-4

Mask - OR of as many of the following bits as are required:

- 0x0800** Pattern set
- 0x0100** Pattern symbol
- 0x0080** Pattern reference point
- 0x0008** Pattern color
- 0x0004** Pattern background color
- 0x0002** Pattern mix
- 0x0001** Pattern background mix

5

Flags

**0x0F** Set indicated default attributes to initial values.  
(Data field is not present in this instance)

**0x8F** Set indicated default attributes to specified values.

6-n

Data - data values as required, in this order if present.

No space is reserved for attributes for which the corresponding Mask flag was not set.

**(1 byte)** - Pattern set

**(1 byte)** - Pattern symbol

**(2\*G bytes)** - Pattern reference point

**(4 bytes)** - Pattern color

**(4 bytes)** - Pattern background color

**(1 byte)** - Pattern mix

**(1 byte)** - Pattern background mix

(G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)

0

0x21 Set Current Defaults

1

Length of following data

2

Set Default Image Attributes 0x06

3-4

Mask - OR of as many of these bits as are required:

**0x0008** Image color

**0x0004** Image background color

**0x0002** Image mix

**0x0001** Image background mix

5

Flags

**0x0F** Set indicated default attributes to initial values. (Data field is not present in this instance)

**0x8F** Set indicated default attributes to specified values.

6-n

Data - data values as required, in this order if present.

No space is reserved for attributes for which the corresponding Mask flag was not set.

**(4 bytes)** - Image color

**(4 bytes)** - Image background color

**(1 byte)** - Image mix

**(1 byte)** - Image background mix

0

0x21 Set Current Defaults

1

Length of following data

2

Set Default Viewing Window 0x05



- 3-4** Mask - OR of as many of the following bits as are required:  
**0x8000** x left limit  
**0x4000** x right limit  
**0x2000** y bottom limit  
**0x1000** y top limit
- 5** Flags  
**0x0F** Set indicated default attributes to initial values.  
(Data field is not present in this case).  
**0x8F** Set indicated default attributes to specified values.
- 6-n** Data - data values as required, in the following order if present.  
No space is reserved for attributes for which the corresponding Mask flag was not set.  
**(2\*G bytes)** - x left limit  
**(2\*G bytes)** - x right limit  
**(2\*G bytes)** - y bottom limit  
**(2\*G bytes)** - y top limit  
(G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)
- 0** 0x21 Set Current Defaults  
**1** Length of following data  
**2** Set Default Arc Parameters 0x0B
- 3-4** Mask - OR of as many of the following bits as are required:  
**0x8000** P value  
**0x4000** Q value  
**0x2000** R value  
**0x1000** S value
- 5** Flags  
**0x0F** Set indicated default attributes to initial values.  
(Data field is not present in this case).  
**0x8F** Set indicated default attributes to specified values.
- 6-n** Data - data values as required, in the following order if present.  
No space is reserved for attributes for which the corresponding Mask flag was not set.  
**(G bytes)** - P value  
**(G bytes)** - Q value  
**(G bytes)** - R value  
**(G bytes)** - S value  
(G=2 or 4 depending on the Geometrics parameter of Set Default Parameter Format)
- 0** 0x21 Set Current Defaults  
**1** Length of following data  
**2** Set Default Pick Identifier 0x0C
- 3-4** Mask - OR of as many of the following bits as are required:  
**0x8000** Pick identifier

|     |  |
|-----|--|
| 5   | Flags  |
|     | <b>0x0F</b> Set indicated default attributes to initial values.<br>(Data field is not present in this case).   |
|     | <b>0x8F</b> Set indicated default attributes to specified values.  |
| 6-n | Data - data values as required, in the following order if present.<br>No space is reserved for attributes for which the corresponding Mask flag was not set.<br><b>(4 bytes)</b> - Pick identifier |
| 0   | 0xE7 Set Bit-map Identifier  |
| 1   | Length of following data 0x07  |
| 2-3 | Usage Flags 0x8000   |
| 4-7 | Bit-map handle   |
| 8   | Lcid   |

**Graphics Data (GAD): optional, repeating**

**Structured Field Introducer**

|     |                                |
|-----|--------------------------------|
| 0-1 | Length 0xn+9                   |
| 2-4 | GAD 0xD3EEBB                   |
| 5   | Flags 0x00                     |
| 6-7 | Segment sequence number 0x0000 |

**Parameters (maximum length in one structured field is 32759)**

**Graphics Segment (optional, repeating)**

Segment data (including the Begin Segment parameter) can be split at any point between successive Graphics Data structured fields.

|       |  |
|-------|--|
| 0     | 0x70 Begin Segment   |
| 1     | Length of following data 0x0E  |
| 2-5   | Segment identifier   |
| 6     | Segment attributes (1)   |
|       | <b>0 B'1'</b> Invisible  |
|       | <b>1 B'1'</b> Propagate invisibility   |
|       | <b>2 B'1'</b> Detectable   |
|       | <b>3 B'1'</b> Propagate detectability  |
|       | <b>6 B'1'</b> Dynamic  |
|       | <b>7 B'1'</b> Fast chaining  |
| 7     | Segment attributes (2)   |
|       | <b>0 B'1'</b> Non-chained  |
|       | <b>3 B'1'</b> Prolog   |
| 8-9   | Segment data length (low-order 2 bytes)  |
| 10-13 | Reserved   |
| 14-15 | Segment data length (high-order 2 bytes)   |
| 16-n  | Graphics orders (see the <i>Graphics Programming Interface Programming Reference</i> ) |

## **End Graphics Object (EGR)**

### **Structured Field Introducer**

0-1 Length 0x0010  
2-4 EGR 0xD3A9BB  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### **Parameters**

0-7 Graphics object name C'0000 0007'

## **End Resource Group (ERG): required**

### **Structured Field Introducer**

0-1 Length 0x0010  
2-4 ERG 0xD3A9C6  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### **Parameters**

0-7 Resource Group name C'0000 0002'

## **End Document (EDT): required**

### **Structured Field Introducer**

0-1 Length 0x0010  
2-4 EDT 0xD3A9A8  
5 Flags 0x00  
6-7 Segment sequence number 0x0000

### **Parameters**

0-7 Document name C'0000 0001'

---

## Appendix G. Initialization File Information

Initialization files include information about printers, queues, and system preferences set by the user from the control panel. Applications can query this information by using the PrfQueryProfileData, PrfQueryProfileInt, PrfQueryProfileSize, and PrfQueryProfileString functions.

All data in initialization files is accessed by a two-level hierarchy of application name, and key name within an application. Presentation Manager system data is keyed off "applications" that have names starting with PM\_.

The application name/key name combinations that applications may need to use are listed below, together with the definition of the corresponding data.

**Note:** Information that is prefixed with PM\_SPOOLERxxxx can not always be modified directly: The spooler validates all attempts to write information to the INI file that it depends on.

|                         |   |
|-------------------------|---|
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "Beep"  |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | 1 or 0.   |
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "LogoDisplayTime"   |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | -1 ≤ time ≤ 32767 milliseconds.<br><b>Indefinite display</b> -1<br><b>No display</b> 0<br><b>Timed display</b> >0 |
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "cxDoubleClick"   |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | SV_CXDBLCLK size in pels.   |
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "cyDoubleClick"   |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | SV_CYDBLCLK size in pels.   |
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "cxMotionStart"   |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | SV_CXMOTIONSTART size in pels.  |
| <b>Application name</b> | "PM_ControlPanel"   |
| <b>Key name</b>         | "cyMotionStart"   |
| <b>Type</b>             | integer   |
| <b>Content/value</b>    | SV_CYMOTIONSTART size in pels.  |

|                         |                        |     |
|-------------------------|------------------------|-----|
| <b>Application name</b> | "PM_National"          |     |
| <b>Key name</b>         | "iCountry"             |     |
| <b>Type</b>             | integer                |     |
| <b>Content/value</b>    | country code:          |     |
|                         | <b>Arabic</b>          | 785 |
|                         | <b>Australian</b>      | 61  |
|                         | <b>Belgian</b>         | 32  |
|                         | <b>Canadian-French</b> | 2   |
|                         | <b>Danish</b>          | 45  |
|                         | <b>Finnish</b>         | 358 |
|                         | <b>French</b>          | 33  |
|                         | <b>German</b>          | 49  |
|                         | <b>Hebrew</b>          | 972 |
|                         | <b>Italian</b>         | 39  |
|                         | <b>Japanese</b>        | 81  |
|                         | <b>Korean</b>          | 82  |
|                         | <b>Latin-American</b>  | 3   |
|                         | <b>Netherlands</b>     | 31  |
|                         | <b>Norwegian</b>       | 47  |
|                         | <b>Portuguese</b>      | 351 |
|                         | <b>Simpl. Chinese</b>  | 86  |
|                         | <b>Spanish</b>         | 34  |
|                         | <b>Swedish</b>         | 46  |
|                         | <b>Swiss</b>           | 41  |
|                         | <b>Trad. Chinese</b>   | 88  |
|                         | <b>UK-English</b>      | 44  |
|                         | <b>US-English</b>      | 1   |
|                         | <b>Other country</b>   | 0.  |

|                         |                      |  |
|-------------------------|----------------------|--|
| <b>Application name</b> | "PM_National"        |  |
| <b>Key name</b>         | "iDate"              |  |
| <b>Type</b>             | integer              |  |
| <b>Content/value</b>    | 0=MDY; 1=DMY; 2=YMD. |  |

|                         |                                     |                                |
|-------------------------|-------------------------------------|--------------------------------|
| <b>Application name</b> | "PM_National"                       |                                |
| <b>Key name</b>         | "iCurrency"                         |                                |
| <b>Type</b>             | integer                             |                                |
| <b>Content/value</b>    | Values have the following meanings: |                                |
|                         | <b>0</b>                            | Prefix, no separator           |
|                         | <b>1</b>                            | Suffix, no separator           |
|                         | <b>2</b>                            | Prefix, 1 character separator  |
|                         | <b>3</b>                            | Suffix, 1 character separator. |

|                         |                               |  |
|-------------------------|-------------------------------|--|
| <b>Application name</b> | "PM_National"                 |  |
| <b>Key name</b>         | "iDigits"                     |  |
| <b>Type</b>             | integer                       |  |
| <b>Content/value</b>    | n = number of decimal digits. |  |

|                         |  |
|-------------------------|--|
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "iTime"                                |
| <b>Type</b>             | integer                                |
| <b>Content/value</b>    | 0 = 12-hour clock; 1 = 24-hour clock.  |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "iLzero"                               |
| <b>Type</b>             | integer                                |
| <b>Content/value</b>    | 0 = no leading zero; 1 = leading zero. |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "s1159"                                |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "am" for example. 3 chars max.         |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "s2359"                                |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "pm" for example. 3 chars max.         |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sCurrency"                            |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "\$" for example. 3 chars max.         |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sThousand"                            |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "," for example. 1 char max.           |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sDecimal"                             |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "." for example. 1 char max.           |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sDate"                                |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "/" for example. 1 char max.           |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sTime"                                |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | ":" for example. 1 char max.           |
| <b>Application name</b> | "PM_National"                          |
| <b>Key name</b>         | "sList"                                |
| <b>Type</b>             | string                                 |
| <b>Content/value</b>    | "," for example. 1 char max.           |

**Application name** PM\_Fonts  
**Key name** <Font module name>  
**Type** string  
**Content/value** fully-qualified drive:\path\filename.ext.

**Application name** "PM\_SPOOLER"  
**Key name** "QUEUE"  
**Type** string  
**Content/value** <Queue name>;

where: <Queue name> is the name of the default queue (might be NULL). This must be a key name for the PM\_SPOOLER\_QUEUE application.

**Application name** "PM\_SPOOLER"  
**Key name** "PRINTER"  
**Type** string  
**Content/value** <Printer name>;

where: <Printer name> is the name of the default printer (might be NULL).

**Note:** Use the SpiQueryDevice and SpiQueryQueue functions to retrieve the spooler configuration data.

## Appendix H. Virtual Key Definitions

The PC VKEY set is shown in the following table:

| Symbol                                 | Personal Computer AT Keyboard  | Enhanced Keyboard  |
|--|--|--|
| VK_BUTTON1<br>VK_BUTTON2<br>VK_BUTTON3 | These values are only used to access the up/down and toggled states of the pointing device buttons; they never actually appear in a WM_CHAR message. | These values are only used to access the up/down and toggled states of the pointing device buttons; they never actually appear in a WM_CHAR message. |
| VK_BREAK                               | Ctrl + Scroll Lock   | Ctrl + Pause   |
| VK_BACKSPACE                           | Backspace  | Backspace  |
| VK_TAB                                 | Tab  | Tab  |
| VK_BACKTAB                             | Shift + Tab  | Shift + Tab  |
| VK_NEWLINE                             | Enter  | Enter  |
| VK_SHIFT *                             | Left and Right Shift   | Left and Right Shift   |
| VK_CTRL *                              | Ctrl   | Left and Right Ctrl  |
| VK_ALT *                               | Alt  | Left and Right Alt   |
| VK_ALTGRAF *                           | None   | Alt Graf (if available)  |
| VK_PAUSE                               | Ctrl + Num Lock  | Pause  |
| VK_CAPSLOCK                            | Caps Lock  | Caps Lock  |
| VK_ESC                                 | Esc  | Esc  |
| VK_SPACE *                             | Space  | Space  |
| VK_PAGEUP *                            | Numpad 9   | Pg Up and Numpad 9   |
| VK_PAGEDOWN *                          | Numpad 3   | Pg Dn and Numpad 3   |
| VK_END *                               | Numpad 1   | End and Numpad 1   |
| VK_HOME *                              | Numpad 7   | Home and Numpad 7  |
| VK_LEFT *                              | Numpad 4   | Left and Numpad 4  |
| VK_UP *                                | Numpad 8   | Up and Numpad 8  |
| VK_RIGHT *                             | Numpad 6   | Right and Numpad 6   |
| VK_DOWN *                              | Numpad 2   | Down and Numpad 2  |
| VK_PRINTSCRN                           | Shift + Print Screen   | Print Screen   |
| VK_INSERT *                            | Numpad 0   | Ins and Numpad 0   |
| VK_DELETE *                            | Numpad &bd.  | Del and Numpad &bd.  |
| VK_SCROLLLOCK                          | Scroll Lock  | Scroll Lock  |
| VK_NUMLOCK                             | Num Lock   | Num Lock   |
| VK_ENTER                               | Shift + Enter  | Shift + Enter and Numpad Enter   |
| VK_SYSRQ                               | SysRq  | Alt + Print Screen   |



|           |      |      |
|-----------|------|------|
| VK_F1 *   | F1   | F1   |
| VK_F2 *   | F2   | F2   |
| VK_F3 *   | F3   | F3   |
| VK_F4 *   | F4   | F4   |
| VK_F5 *   | F5   | F5   |
| VK_F6 *   | F6   | F6   |
| VK_F7 *   | F7   | F7   |
| VK_F8 *   | F8   | F8   |
| VK_F9 *   | F9   | F9   |
| VK_F10 *  | F10  | F10  |
| VK_F11 *  | None | F11  |
| VK_F12 *  | None | F12  |
| VK_F13    | None | None |
| VK_F14    | None | None |
| VK_F15    | None | None |
| VK_F16    | None | None |
| VK_F17    | None | None |
| VK_F18    | None | None |
| VK_F19    | None | None |
| VK_F20    | None | None |
| VK_F21    | None | None |
| VK_F22    | None | None |
| VK_F23    | None | None |
| VK_F24    | None | None |
| VK_MENU * | F10  | F10  |

**Notes:**

1. VKEYs marked with an asterisk (\*) are generated irrespective of other shift states (Shift, Ctrl, Alt, and Alt Graf).
2. VK\_CAPSLOCK is **not** generated for any of the Ctrl shift states, for PC-DOS compatibility.
3. Wherever possible, the VK\_ name is derived from the legend on the key top of the 101-key Enhanced PC keyboard.

---

## Appendix I. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood NY 10594, U.S.A.

---

## Trademarks

The following terms, denoted by an asterisk (\*) in this publication, are trademarks of the IBM Corporation in the United States or other countries:

IBM  
Common User Access  
CUA  
OS/2  
Presentation Manager

The following terms, denoted by a double asterisk (\*\*) in this publication, are trademarks of other companies as follows. Other trademarks are trademarks of their respective companies.

|                 |                            |
|-----------------|----------------------------|
| Adobe           | Adobe Systems Incorporated |
| C++             | AT&T, Incorporated         |
| Helvetica       | Linotype                   |
| Microsoft       | Microsoft Corporation      |
| Pentium         | Intel Corporation          |
| PostScript      | Adobe Systems Incorporated |
| Times New Roman | Monotype                   |
| Windows         | Microsoft Corporation      |



---

# Glossary

This glossary defines many of the terms used in this book. It includes terms and definitions from the *IBM Dictionary of Computing*, as well as terms specific to the OS/2 operating system and the Presentation Manager. It is not a complete glossary for the entire OS/2 operating system; nor is it a complete dictionary of computer terms.

Other primary sources for these definitions are:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyrighted 1990 by the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. These definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

---

## Glossary Listing

### A

**accelerator.** In SAA Common User Access architecture, a key or combination of keys that invokes an application-defined function.

**accelerator table.** A table used to define which key strokes are treated as *accelerators* and the commands they are translated into.

**access mode.** The manner in which an application gains access to a file it has opened. Examples of access modes are read-only, write-only, and read/write.

**access permission.** All access rights that a user has regarding an object. (I)

**action.** One of a set of defined tasks that a computer performs. Users request the application to perform an action in several ways, such as typing a command, pressing a function key, or selecting the action name from an action bar or menu.

**action bar.** In SAA Common User Access architecture, the area at the top of a window that contains choices that give a user access to actions available in that window.

**action point.** The current position on the screen at which the pointer is pointing. Contrast with *hot spot* and *input focus*.

**active program.** A program currently running on the computer. An active program can be interactive (running and receiving input from the user) or noninteractive (running but not receiving input from the user). See also *interactive program* and *noninteractive program*.

**active window.** The window with which the user is currently interacting.

**address space.** (1) The range of addresses available to a program. (A) (2) The area of virtual storage available for a particular job.

**alphanumeric video output.** Output to the logical video buffer when the video adapter is in text mode and the logical video buffer is addressed by an application as a rectangular array of character cells.

**American National Standard Code for Information Interchange.** The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

**Note:** IBM has defined an extension to ASCII code (characters 128-255).

**anchor.** A window procedure that handles Presentation Manager\* message conversions between an icon procedure and an application.

**anchor block.** An area of Presentation-Manager-internal resources to allocated process or thread that calls WinInitialize.

**anchor point.** A point in a window used by a program designer or by a window manager to position a subsequently appearing window.

**ANSI.** American National Standards Institute.

**APA.** All points addressable.

**API.** Application programming interface.

**application.** A collection of software components used to perform specific types of work on a computer; for example, a payroll application, an airline reservation application, a network application.

**application object.** In SAA Advanced Common User Access architecture, a form that an application provides for a user; for example, a spreadsheet form. Contrast with *user object*.

**application programming interface (API).** A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

**application-modal.** Pertaining to a message box or dialog box for which processing must be completed before further interaction with any other window owned by the same application may take place.

**area.** In computer graphics, a filled shape such as a solid rectangle.

**ASCII.** American National Standard Code for Information Interchange.

**ASCIIZ.** A string of ASCII characters that is terminated with a byte containing the value 0.

**aspect ratio.** In computer graphics, the width-to-height ratio of an area, symbol, or shape.

**asynchronous (ASYNC).** (1) Pertaining to two or more processes that do not depend upon the occurrence of specific events such as common timing signals. (T) (2) Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions. See also *synchronous*.

**atom.** A constant that represents a string. As soon as a string has been defined as an atom, the atom can be used in place of the string to save space. Strings are associated with their respective atoms in an *atom table*. See also *integer atom*.

**atom table.** A table used to relate *atoms* with the strings that they represent. Also in the table is the mechanism by which the presence of a string can be checked.

**atomic operation.** An operation that completes its work on an object before another operation can be performed on the same object.

**attribute.** A characteristic or property that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also *graphics attributes* and *segment attributes*.

**automatic link.** In Information Presentation Facility (IPF), a link that begins a chain reaction at the primary window. When the user selects the primary window, an automatic link is activated to display secondary windows.

**AVIO.** Advanced Video Input/Output.

## B

**Bézier curve.** (1) A mathematical technique of specifying smooth continuous lines and surfaces, which require a starting point and a finishing point with several intermediate points that influence or control the path of the linking curve. Named after Dr. P. Bézier. (2) (D of C) In the AIX Graphics Library, a cubic spline approximation to a set of four control points that passes through the first and fourth control points and that has a continuous slope where two spline segments meet. Named after Dr. P. Bézier.

**background.** (1) In multiprogramming, the conditions under which low-priority programs are executed. Contrast with *foreground*. (2) An active session that is not currently displayed on the screen.

**background color.** The color in which the background of a graphic primitive is drawn.

**background mix.** An attribute that determines how the background of a graphic primitive is combined with the existing color of the graphics presentation space. Contrast with *mix*.

**background program.** In multiprogramming, a program that executes with a low priority. Contrast with *foreground program*.

**bit map.** A representation in memory of the data displayed on an APA device, usually the screen.

**block.** (1) A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or logical records. (T) (2) To record data in a block. (3) A collection of contiguous records recorded as a unit. Blocks are separated by interblock gaps and each block may contain one or more records. (A)

**block device.** A storage device that performs I/O operations on blocks of data called *sectors*. Data on block devices can be randomly accessed. Block devices are designated by a drive letter (for example, C:).

**blocking mode.** A condition set by an application that determines when its threads might block. For example, an application might set the Pipemode parameter for the DosCreateNPIPE function so that

its threads perform I/O operations to the named pipe block when no data is available.

**border.** A visual indication (for example, a separator line or a background color) of the boundaries of a window.

**boundary determination.** An operation used to compute the size of the smallest rectangle that encloses a graphics object on the screen.

**breakpoint.** (1) A point in a computer program where execution may be halted. A breakpoint is usually at the beginning of an instruction where halts, caused by external intervention, are convenient for resuming execution. (T) (2) A place in a program, specified by a command or a condition, where the system halts execution and gives control to the workstation user or to a specified program.

**broken pipe.** When all of the handles that access one end of a pipe have been closed.

**bucket.** One or more fields in which the result of an operation is kept.

**buffer.** (1) A portion of storage used to hold input or output data temporarily. (2) To allocate and schedule the use of buffers. (A)

**button.** A mechanism used to request or initiate an action. See also *barrel buttons*, *bezel buttons*, *mouse button*, *push button*, and *radio button*.

**byte pipe.** Pipes that handle data as byte streams. All unnamed pipes are byte pipes. Named pipes can be byte pipes or message pipes. See *byte stream*.

**byte stream.** Data that consists of an unbroken stream of bytes.

## C

**cache.** A high-speed buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

**cached micro presentation space.** A presentation space from a Presentation-Manager-owned store of micro presentation spaces. It can be used for drawing to a window only, and must be returned to the store when the task is complete.

**CAD.** Computer-Aided Design.

**call.** (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) To transfer control to a procedure, program, routine, or subroutine.

**calling sequence.** A sequence of instructions together with any associated data necessary to execute a call. (T)

**Cancel.** An action that removes the current window or menu without processing it, and returns the previous window.

**cascaded menu.** In the OS/2 operating system, a menu that appears when the arrow to the right of a cascading choice is selected. It contains a set of choices that are related to the cascading choice. Cascaded menus are used to reduce the length of a menu. See also *cascading choice*.

**cascading choice.** In SAA Common User Access architecture, a choice in a menu that, when selected, produces a cascaded menu containing other choices. An arrow (→) appears to the right of the cascading choice.

**CASE statement.** In PM programming, provides the body of a window procedure. There is usually one CASE statement for each message type supported by an application.

**CGA.** Color graphics adapter.

**chained list.** A list in which the data elements may be dispersed but in which each data element contains information for locating the next. (T) Synonymous with *linked list*.

**character.** A letter, digit, or other symbol.

**character box.** In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single character from a character set. See also *character mode*. Contrast with *character cell*.

**character cell.** The physical, rectangular space in which any single character is displayed on a screen or printer device. Position is addressed by row and column coordinates. Contrast with *character box*.

**character code.** The means of addressing a character in a character set, sometimes called *code point*.

**character device.** A device that performs I/O operations on one character at a time. Because character devices view data as a stream of bytes, character-device data cannot be randomly accessed. Character devices include the keyboard, mouse, and printer, and are referred to by name.

**character mode.** A mode that, in conjunction with the font type, determines the extent to which graphics characters are affected by the character box, shear, and angle attributes.

**character set.** (1) An ordered set of unique representations called characters; for example, the 26 letters of English alphabet, Boolean 0 and 1, the set of symbols in the Morse code, and the 128 ASCII characters. (A) (2) All the valid characters for a programming language or for a computer system. (3) A group of characters used for a specific reason; for example, the set of characters a printer can print.

**check box.** In SAA Advanced Common User Access architecture, a square box with associated text that represents a choice. When a user selects a choice, an X appears in the check box to indicate that the choice is in effect. The user can clear the check box by selecting the choice again. Contrast with *radio button*.

**check mark.** (1) (D of C) In SAA Advanced Common User Access architecture, a (√) symbol that shows that a choice is currently in effect. (2) The symbol that is used to indicate a selected item on a pull-down menu.

**child process.** In the OS/2 operating system, a process started by another process, which is called the parent process. Contrast with *parent process*.

**child window.** A window that appears within the border of its parent window (either a primary window or another child window). When the parent window is resized, moved, or destroyed, the child window also is resized, moved, or destroyed; however, the child window can be moved or resized independently from the parent window, within the boundaries of the parent window. Contrast with *parent window*.

**choice.** (1) An option that can be selected. The choice can be presented as text, as a symbol (number or letter), or as an icon (a pictorial symbol). (2) (D of C) In SAA Common User Access architecture, an item that a user can select.

**chord.** (1) To press more than one button on a pointing device while the pointer is within the limits that the user has specified for the operating environment. (2) (D of C) In graphics, a short line segment whose end points lie on a circle. Chords are a means for producing a circular image from straight lines. The higher the number of chords per circle, the smoother the circular image.

**class.** In object-oriented design or programming, a group of objects that share a common definition and that therefore share common properties, operations, and behavior. Members of the group are called instances of the class.

**class method.** In System Object Model, an action that can be performed on a class object. Synonymous with factory method.

**class object.** In System Object Model, the run-time implementation of a class.

**class style.** The set of properties that apply to every window in a window class.

**client.** (1) A functional unit that receives shared services from a server. (T) (2) A user, as in a client process that uses a named pipe or queue that is created and owned by a server process.

**client area.** The part of the window, inside the border, that is below the menu bar. It is the user's work space, where a user types information and selects choices from selection fields. In primary windows, it is where an application programmer presents the objects that a user works on.

**client program.** An application that creates and manipulates instances of classes.

**client window.** The window in which the application displays output and receives input. This window is located inside the frame window, under the window title bar and any menu bar, and within any scroll bars.

**clip limits.** The area of the paper that can be reached by a printer or plotter.

**clipboard.** In SAA Common User Access architecture, an area of computer memory, or storage, that temporarily holds data. Data in the clipboard is available to other applications.

**clipping.** In computer graphics, removing those parts of a display image that lie outside a given boundary. (I) (A)

**clipping area.** The area in which the window can paint.

**clipping path.** A clipping boundary in world-coordinate space.

**clock tick.** The minimum unit of time that the system tracks. If the system timer currently counts at a rate of X Hz, the system tracks the time every 1/X of a second. Also known as *time tick*.

**CLOCK\$.** Character-device name reserved for the system clock.

**code page.** An assignment of graphic characters and control-function meanings to all code points.

**code point.** (1) Synonym for *character code*. (2) (D of C) A 1-byte code representing one of 256 potential characters.

**code segment.** An executable section of programming code within a load module.

**color dithering.** See *dithering*.

**color graphics adapter (CGA).** An adapter that simultaneously provides four colors and is supported by all IBM Personal Computer and Personal System/2 models.

**command.** The name and parameters associated with an action that a program can perform.

**command area.** An area composed of a command field prompt and a command entry field.

**command entry field.** An entry field in which users type commands.

**command line.** On a display screen, a display line, sometimes at the bottom of the screen, in which only commands can be entered.

**command mode.** A state of a system or device in which the user can enter commands.

**command prompt.** A field prompt showing the location of the command entry field in a panel.

**Common Programming Interface (CPI).** Definitions of those application development



languages and services that have, or are intended to have, implementations on and a high degree of commonality across the SAA environments. One of the three SAA architectural areas. See also *Common User Access architecture*.

**Common User Access (CUA) architecture.**

Guidelines for the dialog between a human and a workstation or terminal. One of the three SAA architectural areas. See also *Common Programming Interface*.

**compile.** To translate a program written in a higher-level programming language into a machine language program.

**composite window.** A window composed of other windows (such as a frame window, frame-control windows, and a client window) that are kept together as a unit and that interact with each other.

**computer-aided design (CAD).** The use of a computer to design or change a product, tool, or machine, such as using a computer for drafting or illustrating.

**COM1, COM2, COM3.** Character-device names reserved for serial ports 1 through 3.

**CON.** Character-device name reserved for the console keyboard and screen.

**container.** In SAA Common User Access architecture, an object that holds other objects. A folder is an example of a container object. See also *folder* and *object*.

**contextual help.** In SAA Common User Access Architecture, help that gives specific information about the item the cursor is on. The help is contextual because it provides information about a specific item as it is currently being used. Contrast with *extended help*.

**contiguous.** Touching or joining at a common edge or boundary, for example, an unbroken consecutive series of storage locations.

**control.** In SAA Advanced Common User Access architecture, a component of the user interface that allows a user to select choices or type information; for example, a check box, an entry field, a radio button.

**control area.** A storage area used by a computer program to hold control information. (I) (A)

**Control Panel.** In the Presentation Manager, a program used to set up user preferences that act globally across the system.

**Control Program.** (1) The basic functions of the operating system, including DOS emulation and the support for keyboard, mouse, and video input/output. (2) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A)

**control window.** A window that is used as part of a composite window to perform simple input and output tasks. Radio buttons and check boxes are examples.

**control word.** An instruction within a document that identifies its parts or indicates how to format the document.

**coordinate space.** A two-dimensional set of points used to generate output on a video display or printer.

**Copy.** A choice that places onto the clipboard, a copy of what the user has selected. See also *Cut* and *Paste*.

**correlation.** The action of determining which element or object within a picture is at a given position on the display. This follows a *pick* operation.

**coverpage window.** A window in which the application's help information is displayed.

**CPI.** Common Programming Interface.

**critical extended attribute.** An extended attribute that is necessary for the correct operation of the system or a particular application.

**critical section.** (1) In programming languages, a part of an asynchronous procedure that cannot be executed simultaneously with a certain part of another asynchronous procedure. (I)

**Note:** Part of the other asynchronous procedure also is a critical section. (2) A section of code that is not reentrant; that is, code that can be executed by only one thread at a time.

**CUA architecture.** Common User Access architecture.

**current position.** In computer graphics, the position, in user coordinates, that becomes the starting point for the next graphics routine, if that routine does not explicitly specify a starting point.

**cursor.** A symbol displayed on the screen and associated with an input device. The cursor indicates where input from the device will be placed. Types of cursors include text cursors, graphics cursors, and selection cursors. Contrast with *pointer* and *input focus*.

**Cut.** In SAA Common User Access architecture, a choice that removes a selected object, or a part of an object, to the clipboard, usually compressing the space it occupied in a window. See also *Copy* and *Paste*.

## D

**daisy chain.** A method of device interconnection for determining interrupt priority by connecting the interrupt sources serially.

**data segment.** A nonexecutable section of a program module; that is, a section of a program that contains data definitions.

**data structure.** The syntactic structure of symbolic expressions and their storage-allocation characteristics. (T)

**data transfer.** The movement of data from one object to another by way of the clipboard or by direct manipulation.

**DBCS.** Double-byte character set.

**DDE.** Dynamic data exchange.

**deadlock.** (1) Unresolved contention for the use of a resource. (2) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by, or a response from, the other. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

**debug.** To detect, diagnose, and eliminate errors in programs. (T)

**decipoint.** In printing, one tenth of a point. There are 72 points in an inch.

**default procedure.** A function provided by the Presentation Manager Interface that may be used to process standard messages from dialogs or windows.

**default value.** A value assumed when no value has been specified. Synonymous with assumed value. For example, in the graphics programming interface, the default line-type is 'solid'.

**definition list.** A type of list that pairs a term and its description.

**delta.** An application-defined threshold, or number of container items, from either end of the list.

**descendant.** See *child process*.

**descriptive text.** Text used in addition to a field prompt to give more information about a field.

**Deselect all.** A choice that cancels the selection of all of the objects that have been selected in that window.

**Desktop Manager.** In the Presentation Manager, a window that displays a list of groups of programs, each of which can be started or stopped.

**desktop window.** The window, corresponding to the physical device, against which all other types of windows are established.

**detached process.** A background process that runs independent of the parent process.

**detent.** A point on a slider that represents an exact value to which a user can move the slider arm.

**device context.** A logical description of a data destination such as memory, metafile, display, printer, or plotter. See also *direct device context*, *information device context*, *memory device context*, *metafile device context*, *queued device context*, and *screen device context*.

**device driver.** A file that contains the code needed to attach and use a device such as a display, printer, or plotter.

**device space.** (1) Coordinate space in which graphics are assembled after all GPI transformations have been applied. Device space is defined in

device-specific units. (2) (D of C) In computer graphics, a space defined by the complete set of addressable points of a display device. (A)

**dialog.** The interchange of information between a computer and its user through a sequence of requests by the user and the presentation of responses by the computer.

**dialog box.** In SAA Advanced Common User Access architecture, a movable window, fixed in size, containing controls that a user uses to provide information required by an application so that it can continue to process a user request. See also *message box*, *primary window*, *secondary window*. Also known as a *pop-up window*.

**Dialog Box Editor.** A WYSIWYG editor that creates dialog boxes for communicating with the application user.

**dialog item.** A component (for example, a menu or a button) of a dialog box. Dialog items are also used when creating dialog templates.

**dialog procedure.** A dialog window that is controlled by a window procedure. It is responsible for responding to all messages sent to the dialog window.

**dialog tag language.** A markup language used by the DTL compiler to create dialog objects.

**dialog template.** The definition of a dialog box, which contains details of its position, appearance, and window ID, and the window ID of each of its child windows.

**direct device context.** A logical description of a data destination that is a device other than the screen (for example, a printer or plotter), and where the output is not to go through the spooler. Its purpose is to satisfy queries. See also *device context*.

**direct manipulation.** The action of using the mouse to move objects around the screen. For example, moving files and directories around in the *Workplace Shell*.

**direct memory access (DMA).** A technique for moving data directly between main storage and peripheral equipment without requiring processing of the data by the processing unit.(T)

**directory.** A type of file containing the names and controlling information for other files or other directories.

**display point.** Synonym for *pel*.

**dithering.** (1) The process used in color displays whereby every other pel is set to one color, and the intermediate pels are set to another. Together they produce the effect of a third color at normal viewing distances. This process can only be used on solid areas of color; it does not work, for example, on narrow lines. (2) (D of C) In computer graphics, a technique of interleaving dark and light pixels so that the resulting image looks smoothly shaded when viewed from a distance.

**DMA.** Direct memory access.

**DOS Protect Mode Interface (DPMI).** An interface between protect mode and real mode programs.

**double-byte character set (DBCS).** A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more characters than can be represented by 256 code points, require double-byte character sets. Since each character requires two bytes, the entering, displaying, and printing of DBCS characters requires hardware and software that can support DBCS.

**doubleword.** A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit. (A)

**DPMI.** DOS Protect Mode Interface.

**drag.** In SAA Common User Access, to use a pointing device to move an object; for example, clicking on a window border, and dragging it to make the window larger.

**dragging.** (1) In computer graphics, moving an object on the display screen as if it were attached to the pointer. (2) (D of C) In computer graphics, moving one or more segments on a display surface by translating. (I) (A)

**drawing chain.** See *segment chain*.

**drop.** To fix the position of an object that is being dragged, by releasing the select button of the pointing device.

**drop.** To fix the position of an object that is being dragged, by releasing the select button of the pointing device. See also *drag*.

**DTL.** Dialog tag language.

**dual-boot function.** A feature of the OS/2 operating system that allows the user to start DOS from within the operating system, or an OS/2 session from within DOS.

**duplex.** Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex*.

**dynamic data exchange (DDE).** A message protocol used to communicate between applications that share data. The protocol uses shared memory as the means of exchanging data between applications.

**dynamic data formatting.** A formatting procedure that enables you to incorporate text, bit maps or metafiles in an IPF window at execution time.

**dynamic link library.** A collection of executable programming code and data that is bound to an application at load time or run time, rather than during linking. The programming code and data in a dynamic link library can be shared by several applications simultaneously.

**dynamic linking.** The process of resolving external references in a program module at load time or run time rather than during linking.

**dynamic segments.** Graphics segments drawn in exclusive-OR mix mode so that they can be moved from one screen position to another without affecting the rest of the displayed picture.

**dynamic storage.** (1) A device that stores data in a manner that permits the data to move or vary with time such that the specified data is not always available for recovery. (A) (2) A storage in which the cells require repetitive application of control signals in order to retain stored data. Such repetitive application of the control signals is called a refresh operation. A dynamic storage may use static addressing or sensing circuits. (A) (3) See also *static storage*.

**dynamic time slicing.** Varies the size of the time slice depending on system load and paging activity.

**dynamic-link module.** A module that is linked at load time or run time.

## E

**EBCDIC.** Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters (9 bits including parity check), used for information interchange among data processing systems, data communications systems, and associated equipment.

**edge-triggered.** Pertaining to an event semaphore that is posted then reset before a waiting thread gets a chance to run. The semaphore is considered to be posted for the rest of that thread's waiting period; the thread does not have to wait for the semaphore to be posted again.

**EGA.** Extended graphics adapter.

**element.** An entry in a graphics segment that comprises one or more graphics orders and that is addressed by the element pointer.

**EMS.** Expanded Memory Specification.

**encapsulation.** Hiding an object's implementation, that is, its private, internal data and methods. Private variables and methods are accessible only to the object that contains them.

**entry field.** In SAA Common User Access architecture, an area where a user types information. Its boundaries are usually indicated. See also *selection field*.

**entry panel.** A defined panel type containing one or more entry fields and protected information such as headings, prompts, and explanatory text.

**entry-field control.** The component of a user interface that provides the means by which the application receives data entered by the user in an entry field. When it has the input focus, the entry field displays a flashing pointer at the position where the next typed character will go.

**environment segment.** The list of environment variables and their values for a process.

**environment strings.** ASCII text strings that define the value of environment variables.

**environment variables.** Variables that describe the execution environment of a process. These variables are named by the operating system or by the application. Environment variables named by the operating system are PATH, DPATH, INCLUDE, INIT, LIB, PROMPT, and TEMP. The values of environment variables are defined by the user in the CONFIG.SYS file, or by using the SET command at the OS/2 command prompt.

**error message.** An indication that an error has been detected. (A)

**event semaphore.** A semaphore that enables a thread to signal a waiting thread or threads that an event has occurred or that a task has been completed. The waiting threads can then perform an action that is dependent on the completion of the signaled event.

**exception.** An abnormal condition such as an I/O error encountered in processing a data set or a file.

**exclusive system semaphore.** A system semaphore that can be modified only by threads within the same process.

**executable file.** (1) A file that contains programs or commands that perform operations or actions to be taken. (2) A collection of related data records that execute programs.

**exit.** To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on. (T) Repeated exit requests return the user to the point from which all functions provided to the system are accessible. Contrast with *cancel*.

**expanded memory specification (EMS).** Enables DOS applications to access memory above the 1MB real mode addressing limit.

**extended attribute.** An additional piece of information about a file object, such as its data format or category. It consists of a name and a value. A file object may have more than one extended attribute associated with it.

**extended help.** In SAA Common User Access architecture, a help action that provides information about the contents of the application window from which a user requested help. Contrast with *contextual help*.

**extended-choice selection.** A mode that allows the user to select more than one item from a window. Not all windows allow extended choice selection. Contrast with *multiple-choice selection*.

**extent.** Continuous space on a disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

**external link.** In Information Presentation Facility, a link that connects external online document files.

## F

**family-mode application.** An application program that can run in the OS/2 environment and in the DOS environment; however, it cannot take advantage of many of the OS/2-mode facilities, such as multitasking, interprocess communication, and dynamic linking.

**FAT.** File allocation table.

**FEA.** Full extended attribute.

**field-level help.** Information specific to the field on which the cursor is positioned. This help function is "contextual" because it provides information about a specific item as it is currently used; the information is dependent upon the context within the work session.

**FIFO.** First-in-first-out. (A)

**file.** A named set of records stored or processed as a unit. (T)

**file allocation table (FAT).** In IBM personal computers, a table used by the operating system to allocate space on a disk for a file, and to locate and chain together parts of the file that may be scattered on different sectors so that the file can be used in a random or sequential manner.

**file attribute.** Any of the attributes that describe the characteristics of a file.

**File Manager.** In the Presentation Manager, a program that displays directories and files, and allows various actions on them.

**file specification.** The full identifier for a file, which includes its drive designation, path, file name, and extension.

**file system.** The combination of software and hardware that supports storing information on a storage device.

**file system driver (FSD).** A program that manages file I/O and controls the format of information on the storage media.

**fillet.** A curve that is tangential to the end points of two adjoining lines. See also *polyfillet*.

**filtering.** An application process that changes the order of data in a queue.

**first-in-first-out (FIFO).** A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

**flag.** (1) An indicator or parameter that shows the setting of a switch. (2) A character that signals the occurrence of some condition, such as the end of a word. (A) (3) (D of C) A characteristic of a file or directory that enables it to be used in certain ways. See also *archive flag*, *hidden flag*, and *read-only flag*.

**focus.** See *input focus*.

**folder.** A container used to organize objects.

**font.** A particular size and style of typeface that contains definitions of character sets, marker sets, and pattern sets.

**Font Editor.** A utility program provided with the IBM Developers Toolkit that enables the design and creation of new fonts.

**foreground program.** (1) The program with which the user is currently interacting. Also known as *interactive program*. Contrast with *background program*. (2) (D of C) In multiprogramming, a high-priority program.

**frame.** The part of a window that can contain several different visual elements specified by the application, but drawn and controlled by the Presentation Manager. The frame encloses the client area.

**frame styles.** Standard window layouts provided by the Presentation Manager.

**FSD.** File system driver.

**full-duplex.** Synonym for *duplex*.

**full-screen application.** An application that has complete control of the screen.

**function.** (1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a call. (2) A set of related control statements that cause one or more programs to be performed.

**function key.** A key that causes a specified sequence of operations to be performed when it is pressed, for example, F1 and Alt-K.

**function key area.** The area at the bottom of a window that contains function key assignments such as F1=Help.

## G

**GDT.** Global Descriptor Table.

**general protection fault.** An exception condition that occurs when a process attempts to use storage or a module that has some level of protection assigned to it, such as I/O privilege level. See also *IOPL code segment*.

**Global Descriptor Table (GDT).** A table that defines code and data segments available to all tasks in an application.

**global dynamic-link module.** A dynamic-link module that can be shared by all processes in the system that refer to the module name.

**global file-name character.** Either a question mark (?) or an asterisk (\*) used as a variable in a file name or file name extension when referring to a particular file or group of files.

**glyph.** A graphic symbol whose appearance conveys information.

**GPI.** Graphics programming interface.

**graphic primitive.** In computer graphics, a basic element, such as an arc or a line, that is not made up of smaller parts and that is used to create diagrams and pictures. See also *graphics segment*.

**graphics.** (1) A picture defined in terms of graphic primitives and graphics attributes. (2) (D of C) The making of charts and pictures. (3) Pertaining to

charts, tables, and their creation. (4) See *computer graphics, coordinate graphics, fixed-image graphics, interactive graphics, passive graphics, raster graphics*.

**graphics attributes.** Attributes that apply to graphic primitives. Examples are color, line type, and shading-pattern definition. See also *segment attributes*.

**graphics field.** The clipping boundary that defines the visible part of the presentation-page contents.

**graphics mode.** One of several states of a display. The mode determines the resolution and color content of the screen.

**graphics model space.** The conceptual coordinate space in which a picture is constructed after any model transforms have been applied. Also known as *model space*.

**Graphics programming interface.** The formally defined programming language that is between an IBM graphics program and the user of the program.

**graphics segment.** A sequence of related graphic primitives and graphics attributes. See also *graphic primitive*.

**graying.** The indication that a choice on a pull-down is unavailable.

**group.** A collection of logically connected controls. For example, the buttons controlling paper size for a printer could be called a group. See also *program group*.

## H

**handle.** (1) An identifier that represents an object, such as a device or window, to the Presentation Interface. (2) (D of C) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened.

**hard error.** An error condition on a network that requires either that the system be reconfigured or that the source of the error be removed before the system can resume reliable operation.

**header.** (1) System-defined control information that precedes user data. (2) The portion of a message

that contains control information for the message, such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

**heading tags.** A document element that enables information to be displayed in windows, and that controls entries in the contents window controls placement of push buttons in a window, and defines the shape and size of windows.

**heap.** An area of free storage available for dynamic allocation by an application. Its size varies according to the storage requirements of the application.

**help function.** (1) A function that provides information about a specific field, an application panel, or information about the help facility. (2) (D of C) One or more display images that describe how to use application software or how to do a system operation.

**Help index.** In SAA Common User Access architecture, a help action that provides an index of the help information available for an application.

**help panel.** A panel with information to assist users that is displayed in response to a help request from the user.

**help window.** A Common-User-Access-defined secondary window that displays information when the user requests help.

**hidden file.** An operating system file that is not displayed by a directory listing.

**hide button.** In the OS/2 operating system, a small, square button located in the right-hand corner of the title bar of a window that, when selected, removes from the screen all the windows associated with that window. Contrast with *maximize button*. See also *restore button*.

**hierarchical inheritance.** The relationship between parent and child classes. An object that is lower in the inheritance hierarchy than another object, inherits all the characteristics and behaviors of the objects above it in the hierarchy.

**hierarchy.** A tree of segments beginning with the root segment and proceeding downward to dependent segment types.

**high-performance file system (HPFS).** In the OS/2 operating system, an installable file system that uses high-speed buffer storage, known as a cache, to provide fast access to large disk volumes. The file system also supports the coexistence of multiple, active file systems on a single personal computer, with the capability of multiple and different storage devices. File names used with the HPFS can have as many as 254 characters.

**hit testing.** The means of identifying which window is associated with which input device event.

**hook.** A point in a system-defined function where an application can supply additional code that the system processes as though it were part of the function.

**hook chain.** A sequence of hook procedures that are "chained" together so that each event is passed, in turn, to each procedure in the chain.

**hot spot.** The part of the pointer that must touch an object before it can be selected. This is usually the tip of the pointer. Contrast with *action point*.

**HPFS.** high-performance file system.

**hypergraphic link.** A connection between one piece of information and another through the use of graphics.

**hypertext.** A way of presenting information online with connections between one piece of information and another, called *hypertext links*. See also *hypertext link*.

**hypertext link.** A connection between one piece of information and another.

## I

**I/O operation.** An input operation to, or output operation from a device attached to a computer.

**I-beam pointer.** A pointer that indicates an area, such as an entry field in which text can be edited.

**icon.** In SAA Advanced Common User Access architecture, a graphical representation of an object, consisting of an image, image background, and a label. Icons can represent items (such as a document file) that the user wants to work on, and actions that the user wants to perform. In the

Presentation Manager, icons are used for data objects, system actions, and minimized programs.

**icon area.** In the Presentation Manager, the area at the bottom of the screen that is normally used to display the icons for minimized programs.

**Icon Editor.** The Presentation Manager-provided tool for creating icons.

**image font.** A set of symbols, each of which is described in a rectangular array of pels. Some of the pels in the array are set to produce the image of one of the symbols. Contrast with *outline font*.

**indirect manipulation.** Interaction with an object through choices and controls.

**information device context.** A logical description of a data destination other than the screen (for example, a printer or plotter), but where no output will occur. Its purpose is to satisfy queries. See also *device context*.

**information panel.** A defined panel type characterized by a body containing only protected information.

**Information Presentation Facility (IPF).** A facility provided by the OS/2 operating system, by which application developers can produce online documentation and context-sensitive online help panels for their applications.

**input focus.** (1) The area of a window where user interaction is possible using an input device, such as a mouse or the keyboard. (2) The position in the *active window* where a user's normal interaction with the keyboard will appear.

**input router.** An internal OS/2 process that removes messages from the system queue.

**input/output control.** A device-specific command that requests a function of a device driver.

**installable file system (IFS).** A file system in which software is installed when the operating system is started.

**instance.** A single occurrence of an object class that has a particular behavior.

**instruction pointer.** In system/38, a pointer that provides addressability for a machine interface instruction in a program.



**integer atom.** An *atom* that represents a predefined system constant and carries no storage overhead. For example, names of window classes provided by Presentation Manager are expressed as integer atoms.

**interactive graphics.** Graphics that can be moved or manipulated by a user at a terminal.

**interactive program.** (1) A program that is running (active) and is ready to receive (or is receiving) input from a user. (2) A running program that can receive input from the keyboard or another input device. Compare with *active program* and contrast with *noninteractive program*.

Also known as a *foreground program*.

**interchange file.** A file containing data that can be sent from one Presentation Manager interface application to another.

**interpreter.** A program that translates and executes each instruction of a high-level programming language before it translates and executes.

**interprocess communication (IPC).** In the OS/2 operating system, the exchange of information between processes or threads through semaphores, pipes, queues, and shared memory.

**interval timer.** (1) A timer that provides program interruptions on a program-controlled basis. (2) An electronic counter that counts intervals of time under program control.

**IOCtl.** Input/output control.

**IOPL.** Input/output privilege level.

**IOPL code segment.** An IOPL executable section of programming code that enables an application to directly manipulate hardware interrupts and ports without replacing the device driver. See also *privilege level*.

**IPC.** Interprocess communication.

**IPF.** Information Presentation Facility.

**IPF compiler.** A text compiler that interprets tags in a source file and converts the information into the specified format.

**IPF tag language.** A markup language that provides the instructions for displaying online information.

**item.** A data object that can be passed in a DDE transaction.

## J

**journal.** A special-purpose file that is used to record changes made in the system.

## K

**Kanji.** A graphic character set used in Japanese ideographic alphabets.

**KBD\$.** Character-device name reserved for the keyboard.

**kernel.** The part of an operating system that performs basic functions, such as allocating hardware resources.

**Kerning.** The design of graphics characters so that their character boxes overlap. Used to space text proportionally.

**keyboard accelerator.** A keystroke that generates a command message for an application.

**keyboard augmentation.** A function that enables a user to press a keyboard key while pressing a mouse button.

**keyboard focus.** A temporary attribute of a window. The window that has a keyboard focus receives all keyboard input until the focus changes to a different window.

**Keys help.** In SAA Common User Access architecture, a help action that provides a listing of the application keys and their assigned functions.

## L

**label.** In a graphics segment, an identifier of one or more elements that is used when editing the segment.

**LAN.** local area network.

**language support procedure.** A function provided by the Presentation Manager Interface for applications that do not, or cannot (as in the case of COBOL and FORTRAN programs), provide their own dialog or window procedures.

**lazy drag.** See *pickup and drop*.

**lazy drag set.** See *pickup set*.

**LDT.** In the OS/2 operating system, Local Descriptor Table.

**LIFO stack.** A stack from which data is retrieved in last-in, first-out order.

**linear address.** A unique value that identifies the memory object.

**linked list.** Synonym for *chained list*.

**list box.** In SAA Advanced Common User Access architecture, a control that contains scrollable choices from which a user can select one choice.

**Note:** In CUA architecture, this is a programmer term. The end user term is selection list.

**list button.** A button labeled with an underlined down-arrow that presents a list of valid objects or choices that can be selected for that field.

**list panel.** A defined panel type that displays a list of items from which users can select one or more choices and then specify one or more actions to work on those choices.

**load time.** The point in time at which a program module is loaded into main storage for execution.

**load-on-call.** A function of a linkage editor that allows selected segments of the module to be disk resident while other segments are executing. Disk resident segments are loaded for execution and given control when any entry point that they contain is called.

**local area network (LAN).** (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T)

**Note:** A LAN does not use store and forward techniques. (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network.

**Local Descriptor Table (LDT).** Defines code and data segments specific to a single task.

**lock.** A serialization mechanism by means of which a resource is restricted for use by the holder of the lock.

**logical storage device.** A device that the user can map to a physical (actual) device.

**LPT1, LPT2, LPT3.** Character-device names reserved for parallel printers 1 through 3.

## M

**main window.** The window that is positioned relative to the *desktop window*.

**manipulation button.** The button on a pointing device a user presses to directly manipulate an object.

**map.** (1) A set of values having a defined correspondence with the quantities or values of another set. (I) (A) (2) To establish a set of values having a defined correspondence with the quantities or values of another set. (I)

**marker box.** In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single marker from a marker set.

**marker symbol.** A symbol centered on a point. Graphs and charts can use marker symbols to indicate the plotted points.

**marquee box.** The rectangle that appears during a selection technique in which a user selects objects by drawing a box around them with a pointing device.

**Master Help Index.** In the OS/2 operating system, an alphabetic list of help topics related to using the operating system.

**maximize.** To enlarge a window to its largest possible size.

**media window.** The part of the physical device (display, printer, or plotter) on which a picture is presented.

**memory block.** Part memory within a heap.

**memory device context.** A logical description of a data destination that is a memory bit map. See also *device context*.

**memory management.** A feature of the operating system for allocating, sharing, and freeing main storage.

**memory object.** Logical unit of memory requested by an application, which forms the granular unit of memory manipulation from the application viewpoint.

**menu.** In SAA Advanced Common User Access architecture, an extension of the menu bar that displays a list of choices available for a selected choice in the menu bar. After a user selects a choice in menu bar, the corresponding menu appears. Additional pop-up windows can appear from menu choices.

**menu bar.** In SAA Advanced Common User Access architecture, the area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus.

**menu button.** The button on a pointing device that a user presses to view a pop-up menu associated with an object.

**message.** (1) In the Presentation Manager, a packet of data used for communication between the Presentation Manager interface and Presentation Manager applications (2) In a user interface, information not requested by users but presented to users by the computer in response to a user action or internal process.

**message box.** (1) A dialog window predefined by the system and used as a simple interface for applications, without the necessity of creating dialog-template resources or dialog procedures. (2) (D of C) In SAA Advanced Common User Access architecture, a type of window that shows messages to users. See also *dialog box*, *primary window*, *secondary window*.

**message filter.** The means of selecting which messages from a specific window will be handled by the application.

**message queue.** A sequenced collection of messages to be read by the application.

**message stream mode.** A method of operation in which data is treated as a stream of messages. Contrast with *byte stream*.

**metacharacter.** See *global file-name character*.

**metaclass.** The conjunction of an object and its class information; that is, the information pertaining to the class as a whole, rather than to a single instance of the class. Each class is itself an object, which is an instance of the metaclass.

**metafile.** A file containing a series of attributes that set color, shape and size, usually of a picture or a drawing. Using a program that can interpret these attributes, a user can view the assembled image.

**metafile device context.** A logical description of a data destination that is a metafile, which is used for graphics interchange. See also *device context*.

**metalanguage.** A language used to specify another language. For example, data types can be described using a metalanguage so as to make the descriptions independent of any one computer language.

**mickey.** A unit of measurement for physical mouse motion whose value depends on the mouse device driver currently loaded.

**micro presentation space.** A graphics presentation space in which a restricted set of the GPI function calls is available.

**minimize.** To remove from the screen all windows associated with an application and replace them with an icon that represents the application.

**mix.** An attribute that determines how the foreground of a graphic primitive is combined with the existing color of graphics output. Also known as *foreground mix*. Contrast with *background mix*.

**mixed character string.** A string containing a mixture of one-byte and *Kanji* or Hangeul (two-byte) characters.

**mnemonic.** (1) A method of selecting an item on a pull-down by means of typing the highlighted letter in the menu item. (2) (D of C) In SAA Advanced Common User Access architecture, usually a single character, within the text of a choice, identified by an underscore beneath the character. If all characters in a choice already serve as mnemonics for other choices, another character, placed in parentheses immediately following the choice, can be used. When a user types the mnemonic for a choice, the choice is either selected or the cursor is moved to that choice.

**modal dialog box.** In SAA Advanced Common User Access architecture, a type of movable window, fixed in size, that requires a user to enter information before continuing to work in the application window from which it was displayed. Contrast with *modeless dialog box*. Also known as a *serial dialog box*. Contrast with *parallel dialog box*.

**Note:** In CUA architecture, this is a programmer term. The end user term is pop-up window.

**model space.** See *graphics model space*.

**modeless dialog box.** In SAA Advanced Common User Access architecture, a type of movable window, fixed in size, that allows users to continue their dialog with the application without entering information in the dialog box. Also known as a *parallel dialog box*. Contrast with *modal dialog box*.

**Note:** In CUA architecture, this is a programmer term. The end user term is pop-up window.

**module definition file.** A file that describes the code segments within a load module. For example, it indicates whether a code segment is loadable before module execution begins (preload), or loadable only when referred to at run time (load-on-call).

**mouse.** In SAA usage, a device that a user moves on a flat surface to position a pointer on the screen. It allows a user to select a choice or function to be performed or to perform operations on the screen, such as dragging or drawing lines from one position to another.

**MOUSE\$.** Character-device name reserved for a mouse.

**multiple-choice selection.** In SAA Basic Common User Access architecture, a type of field from which

a user can select one or more choices or select none. See also *check box*. Contrast with *extended-choice selection*.

**multiple-line entry field.** In SAA Advanced Common User Access architecture, a control into which a user types more than one line of information. See also *single-line entry field*.

**multitasking.** The concurrent processing of applications or parts of applications. A running application and its data are protected from other concurrently running applications.

**mutex semaphore.** (Mutual exclusion semaphore). A semaphore that enables threads to serialize their access to resources. Only the thread that currently owns the mutex semaphore can gain access to the resource, thus preventing one thread from interrupting operations being performed by another.

**muxwait semaphore.** (Multiple wait semaphore). A semaphore that enables a thread to wait either for multiple event semaphores to be posted or for multiple mutex semaphores to be released. Alternatively, a muxwait semaphore can be set to enable a thread to wait for any ONE of the event or mutex semaphores in the muxwait semaphore's list to be posted or released.

## N

**named pipe.** A named buffer that provides client-to-server, server-to-client, or full duplex communication between unrelated processes. Contrast with *unnamed pipe*.

**national language support (NLS).** The modification or conversion of a United States English product to conform to the requirements of another language or country. This can include the enabling or retrofitting of a product and the translation of nomenclature, MRI, or documentation of a product.

**nested list.** A list that is contained within another list.

**NLS.** national language support.

**non-8.3 file-name format.** A file-naming convention in which file names can consist of up to 255 characters. See also *8.3 file-name format*.

**noncritical extended attribute.** An extended attribute that is not necessary for the function of an application.

**nondestructive read.** Reading that does not erase the data in the source location. (T)

**noninteractive program.** A running program that cannot receive input from the keyboard or other input device. Compare with *active program*, and contrast with *interactive program*.

**nonretained graphics.** Graphic primitives that are not remembered by the Presentation Manager interface when they have been drawn. Contrast with *retained graphics*.

**null character (NUL).** (1) Character-device name reserved for a nonexistent (dummy) device. (2) (D of C) A control character that is used to accomplish media-fill or time-fill and that may be inserted into or removed from a sequence of characters without affecting the meaning of the sequence; however, the control of equipment or the format may be affected by this character. (I) (A)

**null-terminated string.** A string of (n+1) characters where the (n+1)th character is the 'null' character (0x00) Also known as 'zero-terminated' string and 'ASCIIZ' string.

## O

**object.** A set of data and actions that can be performed on that data.

**Object Interface Definition Language (OIDL).** Specification language for SOM class definitions.

**object window.** A window that does not have a parent but which might have child windows. An object window cannot be presented on a device.

**OIDL.** Object Interface Definition Language.

**open.** To start working with a file, directory, or other object.

**ordered list.** Vertical arrangements of items, with each item in the list preceded by a number or letter.

**outline font.** A set of symbols, each of which is created as a series of lines and curves.

Synonymous with *vector font*. Contrast with *image font*.

**output area.** An area of storage reserved for output. (A)

**owner window.** A window into which specific events that occur in another (owned) window are reported.

**ownership.** The determination of how windows communicate using messages.

**owning process.** The process that owns the resources that might be shared with other processes.

## P

**page.** (1) A 4KB segment of contiguous physical memory. (2) (D of C) A defined unit of space on a storage medium.

**page viewport.** A boundary in device coordinates that defines the area of the output device in which graphics are to be displayed. The presentation-page contents are transformed automatically to the page viewport in device space.

**paint.** (1) The action of drawing or redrawing the contents of a window. (2) In computer graphics, to shade an area of a display image; for example, with crosshatching or color.

**panel.** In SAA Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

**panel area.** An area within a panel that contains related information. The three major Common User Access-defined panel areas are the action bar, the function key area, and the panel body.

**panel area separator.** In SAA Basic Common User Access architecture, a solid, dashed, or blank line that provides a visual distinction between two adjacent areas of a panel.

**panel body.** The portion of a panel not occupied by the action bar, function key area, title or scroll bars. The panel body can contain protected information, selection fields, and entry fields. The layout and content of the panel body determine the panel type.

**panel body area.** See *client area*.

**panel definition.** A description of the contents and characteristics of a panel. A panel definition is the application developer's mechanism for predefining the format to be presented to users in a window.

**panel ID.** In SAA Basic Common User Access architecture, a panel identifier, located in the upper-left corner of a panel. A user can choose whether to display the panel ID.

**panel title.** In SAA Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

**paper size.** The size of paper, defined in either standard U.S. or European names (for example, A, B, A4), and measured in inches or millimeters respectively.

**parallel dialog box.** See *modeless dialog box*.

**parameter list.** A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

**parent process.** In the OS/2 operating system, a process that creates other processes. Contrast with *child process*.

**parent window.** In the OS/2 operating system, a window that creates a child window. The child window is drawn within the parent window. If the parent window is moved, resized, or destroyed, the child window also will be moved, resized, or destroyed. However, the child window can be moved and resized independently from the parent window, within the boundaries of the parent window. Contrast with *child window*.

**partition.** (1) A fixed-size division of storage. (2) On an IBM personal computer fixed disk, one of four possible storage areas of variable size; one may be accessed by DOS, and each of the others may be assigned to another operating system.

**Paste.** A choice in the Edit pull-down that a user selects to move the contents of the clipboard into a preselected location. See also *Copy* and *Cut*.

**path.** The route used to locate files; the storage location of a file. A fully qualified path lists the drive identifier, directory name, subdirectory name (if any), and file name with the associated extension.

**PDD.** Physical device driver.

**peeking.** An action taken by any thread in the process that owns the queue to examine queue elements without removing them.

**pel.** (1) The smallest area of a display screen capable of being addressed and switched between visible and invisible states. Synonym for *display point*, *pixel*, and *picture element*. (2) (D of C) Picture element.

**physical device driver (PDD).** A system interface that handles hardware interrupts and supports a set of input and output functions.

**pick.** To select part of a displayed object using the pointer.

**pickup.** To add an object or set of objects to the pickup set.

**pickup and drop.** A drag operation that does not require the direct manipulation button to be pressed for the duration of the drag.

**pickup set.** The set of objects that have been picked up as part of a pickup and drop operation.

**picture chain.** See *segment chain*.

**picture element.** (1) Synonym for *pel*. (2) (D of C) In computer graphics, the smallest element of a display surface that can be independently assigned color and intensity. (T) (3) The area of the finest detail that can be reproduced effectively on the recording medium.

**PID.** Process identification.

**pipe.** (1) A named or unnamed buffer used to pass data between processes. A process reads from or writes to a pipe as if the pipe were a standard-input or standard-output file. See also *named pipe* and *unnamed pipe*. (2) (D of C) To direct data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (`|`).

**pixel.** (1) Synonym for *pel*. (2) (D of C) Picture element.

**plotter.** An output unit that directly produces a hardcopy record of data on a removable medium, in the form of a two-dimensional graphic representation. (T)

**PM.** Presentation Manager.

**pointer.** (1) The symbol displayed on the screen that is moved by a pointing device, such as a *mouse*. The pointer is used to point at items that users can select. Contrast with *cursor*. (2) A data element that indicates the location of another data element. (T)

**POINTER\$.** Character-device name reserved for a pointer device (mouse screen support).

**pointing device.** In SAA Advanced Common User Access architecture, an instrument, such as a mouse, trackball, or joystick, used to move a pointer on the screen.

**pointings.** Pairs of x-y coordinates produced by an operator defining positions on a screen with a pointing device, such as a *mouse*.

**polyfillet.** A curve based on a sequence of lines. The curve is tangential to the end points of the first and last lines, and tangential also to the midpoints of all other lines. See also *fillet*.

**polygon.** One or more closed figures that can be drawn filled, outlined, or filled and outlined.

**polyline.** A sequence of adjoining lines.

**polymorphism.** A concept whereby the behavior of an application object is dependent solely upon the class and contents of the messages received by that object, and is not affected by any other external factor.

**pop.** To retrieve an item from a last-in-first-out stack of items. Contrast with *push*.

**pop-up window.** (1) A window that appears on top of another window in a dialog. Each pop-up window must be completed before returning to the underlying window. (2) (D of C) In SAA Advanced Common User Access architecture, a movable window, fixed in size, in which a user provides information required by an application so that it can continue to process a user request.

**presentation drivers.** Special purpose I/O routines that handle field device-independent I/O requests from the PM and its applications.

**Presentation Manager (PM).** The interface of the OS/2 operating system that presents, in windows a graphics-based interface to applications and files installed and running under the OS/2 operating system.

**presentation page.** The coordinate space in which a picture is assembled for display.

**presentation space (PS).** (1) Contains the device-independent definition of a picture. (2) (D of C) The display space on a display device.

**primary window.** In SAA Common User Access architecture, the window in which the main interaction between the user and the application takes place. In a multiprogramming environment, each application starts in its own primary window. The primary window remains for the duration of the application, although the panel displayed will change as the user's dialog moves forward. See also *secondary window*.

**primitive.** In computer graphics, one of several simple functions for drawing on the screen, including, for example, the rectangle, line, ellipse, polygon, and so on.

**primitive attribute.** A specifiable characteristic of a graphic primitive. See *graphics attributes*.

**print job.** The result of sending a document or picture to be printed.

**Print Manager.** In the Presentation Manager, the part of the spooler that manages the spooling process. It also allows users to view print queues and to manipulate print jobs.

**privilege level.** A protection level imposed by the hardware architecture of the IBM personal computer. There are four privilege levels (number 0 through 3). Only certain types of programs are allowed to execute at each privilege level. See also *IOPL code segment*.

**procedure call.** In programming languages, a language construct for invoking execution of a procedure.

**process.** An instance of an executing application and the resources it is using.

**program.** A sequence of instructions that a computer can interpret and execute.

**program details.** Information about a program that is specified in the *Program Manager* window and is used when the program is started.

**program group.** In the Presentation Manager, several programs that can be acted upon as a single entity.

**program name.** The full file specification of a program. Contrast with *program title*.

**program title.** The name of a program as it is listed in the *Program Manager* window. Contrast with *program name*.

**prompt.** A displayed symbol or message that requests input from the user or gives operational information; for example, on the display screen of an IBM personal computer, the DOS A> prompt. The user must respond to the prompt in order to proceed.

**protect mode.** A method of program operation that limits or prevents access to certain instructions or areas of storage. Contrast with *real mode*.

**protocol.** A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (I)

**pseudocode.** An artificial language used to describe computer program algorithms without using the syntax of any particular programming language. (A)

**pull-down.** (1) An *action bar* extension that displays a list of choices available for a selected action bar choice. After users select an action bar choice, the pull-down appears with the list of choices. Additional *pop-up windows* may appear from pull-down choices to further extend the actions available to users. (2) (D of C) In SAA Common User Access architecture, pertaining to a choice in an action bar pull-down.

**push.** To add an item to a last-in-first-out stack of items. Contrast with *pop*.

**push button.** In SAA Advanced Common User Access architecture, a rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected.

**putback.** To remove an object or set of objects from the lazy drag set. This has the effect of undoing the pickup operation for those objects

**putdown.** To drop the objects in the lazy drag set on the target object.

## Q

**queue.** (1) A linked list of elements waiting to be processed in FIFO order. For example, a queue may be a list of print jobs waiting to be printed. (2) (D of C) A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed.

**queued device context.** A logical description of a data destination (for example, a printer or plotter) where the output is to go through the spooler. See also *device context*.

## R

**radio button.** (1) A control window, shaped like a round button on the screen, that can be in a checked or unchecked state. It is used to select a single item from a list. Contrast with *check box*. (2) In SAA Advanced Common User Access architecture, a circle with text beside it. Radio buttons are combined to show a user a fixed set of choices from which only one can be selected. The circle is partially filled when a choice is selected.

**RAS.** Reliability, availability, and serviceability.

**raster.** (1) In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. (T) (2) The coordinate grid that divides the display area of a display device. (A)

**read-only file.** A file that can be read from but not written to.

**real mode.** A method of program operation that does not limit or prevent access to any instructions or areas of storage. The operating system loads the entire program into storage and gives the program access to all system resources. Contrast with *protect mode*.

**realize.** To cause the system to ensure, wherever possible, that the physical color table of a device is



set to the closest possible match in the logical color table.

**recursive routine.** A routine that can call itself, or be called by another routine that was called by the recursive routine.

**reentrant.** The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

**reference phrase.** (1) A word or phrase that is emphasized in a device-dependent manner to inform the user that additional information for the word or phrase is available. (2) (D of C) In hypertext, text that is highlighted and preceded by a single-character input field used to signify the existence of a hypertext link.

**reference phrase help.** In SAA Common User Access architecture, highlighted words or phrases within help information that a user selects to get additional information.

**refresh.** To update a window, with changed information, to its current status.

**region.** A clipping boundary in device space.

**register.** A part of internal storage having a specified storage capacity and usually intended for a specific purpose. (T)

**remote file system.** A file-system driver that gains access to a remote system without a block device driver.

**resource.** The means of providing extra information used in the definition of a window. A resource can contain definitions of fonts, templates, accelerators, and mnemonics; the definitions are held in a resource file.

**resource file.** A file containing information used in the definition of a window. Definitions can be of fonts, templates, accelerators, and mnemonics.

**restore.** To return a window to its original size or position following a sizing or moving action.

**retained graphics.** Graphic primitives that are remembered by the Presentation Manager interface after they have been drawn. Contrast with *nonretained graphics*.

**return code.** (1) A value returned to a program to indicate the results of an operation requested by that program. (2) A code used to influence the execution of succeeding instructions.(A)

**reverse video.** (1) A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background. (2) In SAA Basic Common User Access architecture, a screen emphasis feature that interchanges the foreground and background colors of an item.

**REXX Language.** Restructured Extended Executor. A procedural language that provides batch language functions along with structured programming constructs such as loops; conditional testing and subroutines.

**RGB.** (1) Color coding in which the brightness of the additive primary colors of light, red, green, and blue, are specified as three distinct values of white light. (2) Pertaining to a color display that accepts signals representing red, green, and blue.

**roman.** Relating to a type style with upright characters.

**root segment.** In a hierarchical database, the highest segment in the tree structure.

**round-robin scheduling.** A process that allows each thread to run for a specified amount of time.

**run time.** (1) Any instant at which the execution of a particular computer program takes place. (T) (2) The amount of time needed for the execution of a particular computer program. (T) (3) The time during which an instruction in an instruction register is decoded and performed. Synonym for *execution time*.

## S

**SAA.** Systems Application Architecture.

**SBCS.** Single-byte character set.

**scheduler.** A computer program designed to perform functions such as scheduling, initiation, and termination of jobs.

**screen.** In SAA Basic Common User Access architecture, the physical surface of a display device upon which information is shown to a user.

**screen device context.** A logical description of a data destination that is a particular window on the screen. See also *device context*.

**SCREEN\$.** Character-device name reserved for the display screen.

**scroll bar.** In SAA Advanced Common User Access architecture, a part of a window, associated with a scrollable area, that a user interacts with to see information that is not currently allows visible.

**scrollable entry field.** An entry field larger than the visible field.

**scrollable selection field.** A selection field that contains more choices than are visible.

**scrolling.** Moving a display image vertically or horizontally in a manner such that new data appears at one edge, as existing data disappears at the opposite edge.

**secondary window.** A window that contains information that is dependent on information in a primary window and is used to supplement the interaction in the primary window.

**sector.** On disk or diskette storage, an addressable subdivision of a track used to record one block of a program or data.

**segment.** See *graphics segment*.

**segment attributes.** Attributes that apply to the segment as an entity, as opposed to the individual primitives within the segment. For example, the visibility or detectability of a segment.

**segment chain.** All segments in a graphics presentation space that are defined with the 'chained' attribute. Synonym for *picture chain*.

**segment priority.** The order in which segments are drawn.

**segment store.** An area in a normal graphics presentation space where retained graphics segments are stored.

**select.** To mark or choose an item. Note that *select* means to mark or type in a choice on the

screen; *enter* means to send all selected choices to the computer for processing.

**select button.** The button on a pointing device, such as a mouse, that is pressed to select a menu choice. Also known as button 1.

**selection cursor.** In SAA Advanced Common User Access architecture, a visual indication that a user has selected a choice. It is represented by outlining the choice with a dotted box. See also *text cursor*.

**selection field.** (1) In SAA Advanced Common User Access architecture, a set of related choices. See also *entry field*. (2) In SAA Basic Common User Access architecture, an area of a panel that cannot be scrolled and contains a fixed number of choices.

**semantics.** The relationships between symbols and their meanings.

**semaphore.** An object used by applications for signalling purposes and for controlling access to serially reusable resources.

**separator.** In SAA Advanced Common User Access architecture, a line or color boundary that provides a visual distinction between two adjacent areas.

**serial dialog box.** See *modal dialog box*.

**serialization.** The consecutive ordering of items.

**serialize.** To ensure that one or more events occur in a specified sequence.

**serially reusable resource (SRR).** A logical resource or object that can be accessed by only one task at a time.

**session.** (1) A routing mechanism for user interaction via the console; a complete environment that determines how an application runs and how users interact with the application. OS/2 can manage more than one session at a time, and more than one process can run in a session. Each session has its own set of environment variables that determine where OS/2 looks for dynamic-link libraries and other important files. (2) (D of C) In the OS/2 operating system, one instance of a started program or command prompt. Each session is separate from all other sessions that might be running on the computer. The operating system is responsible for coordinating the resources that each

session uses, such as computer memory, allocation of processor time, and windows on the screen.

**Settings Notebook.** A control window that is used to display the settings for an object and to enable the user to change them.

**shadow box.** The area on the screen that follows mouse movements and shows what shape the window will take if the mouse button is released.

**shared data.** Data that is used by two or more programs.

**shared memory.** In the OS/2 operating system, a segment that can be used by more than one program.

**shear.** In computer graphics, the forward or backward slant of a graphics symbol or string of such symbols relative to a line perpendicular to the baseline of the symbol.

**shell.** (1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens, and communicate them to the operating system. (2) Software that allows a kernel program to run under different operating-system environments.

**shutdown.** The process of ending operation of a system or a subsystem, following a defined procedure.

**sibling processes.** Child processes that have the same parent process.

**sibling windows.** Child windows that have the same parent window.

**simple list.** A list of like values; for example, a list of user names. Contrast with *mixed list*.

**single-byte character set (SBCS).** A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set*.

**slider box.** In SAA Advanced Common User Access architecture: a part of the scroll bar that shows the position and size of the visible information in a window relative to the total amount of information available. Also known as *thumb mark*.

**SOM.** System Object Model.

**source file.** A file that contains source statements for items such as high-level language programs and data description specifications.

**source statement.** A statement written in a programming language.

**specific dynamic-link module.** A dynamic-link module created for the exclusive use of an application.

**spin button.** In SAA Advanced Common User Access architecture, a type of entry field that shows a scrollable ring of choices from which a user can select a choice. After the last choice is displayed, the first choice is displayed again. A user can also type a choice from the scrollable ring into the entry field without interacting with the spin button.

**spline.** A sequence of one or more Bézier curves.

**spooler.** A program that intercepts the data going to printer devices and writes it to disk. The data is printed or plotted when it is complete and the required device is available. The spooler prevents output from different sources from being intermixed.

**stack.** A list constructed and maintained so that the next data element to be retrieved is the most recently stored. This method is characterized as last-in-first-out (LIFO).

**standard window.** A collection of window elements that form a panel. The standard window can include one or more of the following window elements: sizing borders, system menu icon, title bar, maximize/minimize/restore icons, action bar and pull-downs, scroll bars, and client area.

**static control.** The means by which the application presents descriptive information (for example, headings and descriptors) to the user. The user cannot change this information.

**static storage.** (1) A read/write storage unit in which data is retained in the absence of control signals. (A) Static storage may use dynamic addressing or sensing circuits. (2) Storage other than *dynamic storage*. (A)

**style.** See *window style*.

**subdirectory.** In an IBM personal computer, a file referred to in a root directory that contains the

names of other files stored on the diskette or fixed disk.

**swapping.** (1) A process that interchanges the contents of an area of real storage with the contents of an area in auxiliary storage. (I) (A) (2) In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage. (3) The process of temporarily removing an active job from main storage, saving it on disk, and processing another job in the area of main storage formerly occupied by the first job.

**switch.** (1) In SAA usage, to move the cursor from one point of interest to another; for example, to move from one screen or window to another or from a place within a displayed image to another place on the same displayed image. (2) In a computer program, a conditional instruction and an indicator to be interrogated by that instruction. (3) A device or programming technique for making a selection, for example, a toggle, a conditional jump.

**switch list.** See *Task List*.

**symbolic identifier.** A text string that equates to an integer value in an include file, which is used to identify a programming object.

**symbols.** In Information Presentation Facility, a document element used to produce characters that cannot be entered from the keyboard.

**synchronous.** Pertaining to two or more processes that depend upon the occurrence of specific events such as common timing signals. (T) See also *asynchronous*.

**System Menu.** In the Presentation Manager, the pull-down in the top left corner of a window that allows it to be moved and sized with the keyboard.

**System Object Model (SOM).** A mechanism for language-neutral, object-oriented programming in the OS/2 environment.

**system queue.** The master queue for all pointer device or keyboard events.

**system-defined messages.** Messages that control the operations of applications and provides input an other information for applications to process.

**Systems Application Architecture (SAA).** A set of IBM software interfaces, conventions, and protocols that provide a framework for designing and developing applications that are consistent across systems.

## T

**table tags.** In Information Presentation Facility, a document element that formats text in an arrangement of rows and columns.

**tag.** (1) One or more characters attached to a set of data that contain information about the set, including its identification. (I) (A) (2) In Generalized Markup Language markup, a name for a type of document or document element that is entered in the source document to identify it.

**target object.** An object to which the user is transferring information.

**Task List.** In the Presentation Manager, the list of programs that are active. The list can be used to switch to a program and to stop programs.

**template.** An ASCII-text definition of an action bar and pull-down menu, held in a resource file, or as a data structure in program memory.

**terminate-and-stay-resident (TSR).** Pertaining to an application that modifies an operating system interrupt vector to point to its own location (known as hooking an interrupt).

**text.** Characters or symbols.

**text cursor.** A symbol displayed in an entry field that indicates where typed input will appear.

**text window.** Also known as the VIO window.

**text-windowed application.** The environment in which the operating system performs advanced-video input and output operations.

**thread.** A unit of execution within a process. It uses the resources of the process.

**thumb mark.** The portion of the scroll bar that describes the range and properties of the data that is currently visible in a window. Also known as a *slider box*.

**thunk.** Term used to describe the process of address conversion, stack and structure realignment, etc., necessary when passing control between 16-bit and 32-bit modules.

**tilde.** A mark used to denote the character that is to be used as a mnemonic when selecting text items within a menu.

**time slice.** (1) An interval of time on the processing unit allocated for use in performing a task. After the interval has expired, processing-unit time is allocated to another task, so a task cannot monopolize processing-unit time beyond a fixed limit. (2) In systems with time sharing, a segment of time allocated to a terminal job.

**time-critical process.** A process that must be performed within a specified time after an event has occurred.

**timer.** A facility provided under the Presentation Manager, whereby Presentation Manager will dispatch a message of class WM\_TIMER to a particular window at specified intervals. This capability may be used by an application to perform a specific processing task at predetermined intervals, without the necessity for the application to explicitly keep track of the passage of time.

**timer tick.** See *clock tick*.

**title bar.** In SAA Advanced Common User Access architecture, the area at the top of each window that contains the window title and system menu icon. When appropriate, it also contains the minimize, maximize, and restore icons. Contrast with *panel title*.

**TLB.** Translation lookaside buffer.

**transaction.** An exchange between a workstation and another device that accomplishes a particular action or result.

**transform.** (1) The action of modifying a picture by scaling, shearing, reflecting, rotating, or translating. (2) The object that performs or defines such a modification; also referred to as a *transformation*.

**Translation lookaside buffer (TLB).** A hardware-based address caching mechanism for paging information.

**Tree.** In the Presentation Manager, the window in the *File Manager* that shows the organization of drives and directories.

**truncate.** (1) To terminate a computational process in accordance with some rule (A) (2) To remove the beginning or ending elements of a string. (3) To drop data that cannot be printed or displayed in the line width specified or available. (4) To shorten a field or statement to a specified length.

**TSR.** Terminate-and-stay-resident.

**unnamed pipe.** A circular buffer, created in memory, used by related processes to communicate with one another. Contrast with *named pipe*.

**unordered list.** In Information Presentation Facility, a vertical arrangement of items in a list, with each item in the list preceded by a special character or bullet.

**update region.** A system-provided area of dynamic storage containing one or more (not necessarily contiguous) rectangular areas of a window that are visually invalid or incorrect, and therefore are in need of repainting.

**user interface.** Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

**User Shell.** A component of OS/2 that uses a graphics-based, windowed interface to allow the user to manage applications and files installed and running under OS/2.

**utility program.** (1) A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. (T) (2) A program designed to perform an everyday task such as copying data from one storage device to another. (A)

## U

There are no glossary terms for this starting letter.

## V

**value set control.** A visual component that enables a user to select one choice from a group of mutually exclusive choices.

**vector font.** A set of symbols, each of which is created as a series of lines and curves. Synonymous with *outline font*. Contrast with *image font*.

**VGA.** Video graphics array.

**viewing pipeline.** The series of transformations applied to a graphic object to map the object to the device on which it is to be presented.

**viewing window.** A clipping boundary that defines the visible part of model space.

**VIO.** Video Input/Output.

**virtual memory (VM).** Synonymous with *virtual storage*.

**virtual storage.** (1) The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations. (l) (A) (2) Addressable space that is apparent to the user as the processor storage space, from which the instructions and the data are mapped into the processor storage locations. (3) Synonymous with *virtual memory*.

**visible region.** A window's presentation space, clipped to the boundary of the window and the boundaries of any overlying window.

**volume.** (1) A file-system driver that uses a block device driver for input and output operations to a local or remote device. (l) (2) A portion of data, together with its data carrier, that can be handled conveniently as a unit.

## W

**wildcard character.** Synonymous with *global file-name character*.

**window.** (1) A portion of a display surface in which display images pertaining to a particular application can be presented. Different applications can be displayed simultaneously in different windows. (A) (2) An area of the screen with visible boundaries within which information is displayed. A window can be smaller than or the same size as the screen. Windows can appear to overlap on the screen.

**window class.** The grouping of windows whose processing needs conform to the services provided by one window procedure.

**window coordinates.** A set of coordinates by which a window position or size is defined; measured in device units, or *pels*.

**window handle.** Unique identifier of a window, generated by Presentation Manager when the window is created, and used by applications to direct messages to the window.

**window procedure.** Code that is activated in response to a message. The procedure controls the appearance and behavior of its associated windows.

**window rectangle.** The means by which the size and position of a window is described in relation to the desktop window.

**window resource.** A read-only data segment stored in the .EXE file of an application or the .DLL file of a dynamic link library.

**window style.** The set of properties that influence how events related to a particular window will be processed.

**window title.** In SAA Advanced Common User Access architecture, the area in the title bar that contains the name of the application and the OS/2 operating system file name, if applicable.

**workstation.** (1) A display screen together with attachments such as a keyboard, a local copy device, or a tablet. (2) (D of C) One or more programmable or nonprogrammable devices that allow a user to do work.

**world coordinates.** A device-independent Cartesian coordinate system used by the application program for specifying graphical input and output. (I) (A)

**world-coordinate space.** Coordinate space in which graphics are defined before transformations are applied.

**WYSIWYG.** What-You-See-Is-What-You-Get. A capability of a text editor to continually display pages exactly as they will be printed.

## X

There are no glossary terms for this starting letter.

## Y

There are no glossary terms for this starting letter.

## Z

**z-order.** The order in which sibling windows are presented. The topmost sibling window obscures any portion of the siblings that it overlaps; the same effect occurs down through the order of lower sibling windows.

**zooming.** The progressive scaling of an entire display image in order to give the visual impression of movement of all or part of a display group toward or away from an observer. (I) (A)

**8.3 file-name format.** A file-naming convention in which file names are limited to eight characters before and three characters after a single dot. Usually pronounced "eight-dot-three." See also *non-8.3 file-name format*.

---

# Index

## A

ACCEL A-1  
ACCELTABLE A-1  
ACCELTABLE statement 31-10  
ACCELTABLE statement 31-9  
APIRET A-2  
Applications  
    Windowed PM 32-1  
APSZ A-2  
ARCPARAMS A-2  
AREABUNDLE A-3  
ASSOCTABLE statement 31-9, 31-12  
ATOM A-4

## B

BDS\_\* values 11-6  
bit maps  
    data D-1  
    example D-2  
    file format D-2  
    information tables D-1  
    standard formats D-1  
BIT16 A-20  
BIT32 A-20  
BIT8 A-20  
BITMAPARRAYFILEHEADER A-4  
BITMAPARRAYFILEHEADER2 A-5  
BITMAPFILEHEADER A-6  
BITMAPFILEHEADER2 A-7  
BITMAPINFO A-8  
BITMAPINFO2 A-9  
BITMAPINFOHEADER A-14  
BITMAPINFOHEADER2 A-15  
BKM\_CALCPIGMENTRECT 23-7  
BKM\_DELETEPAGE 23-8  
BKM\_INSERTPAGE 23-9  
BKM\_INVALIDATETABS 23-11  
BKM\_QUERYPAGECOUNT 23-11  
BKM\_QUERYPAGEDATA 23-13  
BKM\_QUERYPAGEID 23-13  
BKM\_QUERYPAGEINFO 23-15  
BKM\_QUERYPAGESTYLE 23-16  
BKM\_QUERYPAGEWINDOWHWND 23-17  
BKM\_QUERYSTATUSLINETEXT 23-18

BKM\_QUERYTABBITMAP 23-19  
BKM\_QUERYTABTEXT 23-19, 23-20  
BKM\_SETDIMENSIONS 23-20, 23-21  
BKM\_SETNOTEBOOKCOLORS 23-21, 23-22  
BKM\_SETPAGEDATA 23-23  
BKM\_SETPAGEINFO 23-24  
BKM\_SETPAGEWINDOWHWND 23-24, 23-25  
BKM\_SETSTATUSLINETEXT 23-26  
BKM\_SETTABBITMAP 23-26  
BKM\_SETTABTEXT 23-28  
BKM\_TURNTOPAGE 23-28, 23-29  
BKN\_\* values 23-3  
BKS\_\* values 23-1  
BM\_CLICK 11-8  
BM\_QUERYCHECK 11-8  
BM\_QUERYCHECKINDEX 11-9  
BM\_QUERYHILITE 11-10  
BM\_SETCHECK 11-11  
BM\_SETDEFAULT 11-12  
BM\_SETHILITE 11-13  
BN\_\* values 11-6  
BOOKPAGEINFO A-21  
BOOKTEXT A-23  
BOOL A-21  
BS\_\* values 11-1  
BTNCDATA A-24  
button control data 11-3  
button control styles 11-1  
button control window processing 11-1  
BYTE A-24

## C

CA\_\* values A-36  
    column headings A-37  
    drawing and painting A-37  
    icons or bit maps A-36  
    ordered target emphasis A-36  
    title attributes A-37  
    title position A-37  
    titles A-37  
CATCHBUF A-25  
CBM\_HILITE 17-6  
CBM\_ISLISTSHOWING 17-7  
CBM\_SHOWLIST 17-7  
CBN\_\* values 17-3



CBS\_\* values 17-1  
 CCS\_\* values  
     selection types 22-3  
     styles 22-2  
 CDATE A-25  
 CDP parameter A-52  
 CF\_\* values 27-5  
 CFA\_\* values A-73  
     data types A-73  
 CHAR A-26  
 CHARBUNDLE A-26  
 check box 11-1  
 CLASSINFO A-27  
 clipboard 26-27  
     messages 26-27  
 Clipboard messages 26-27, 27-1  
 clipping F-1  
 CM\_ALLOCDETAILFIELDINFO 22-31  
 CM\_ALLOCRECORD 22-32  
 CM\_ARRANGE 22-34  
 CM\_CLOSEEDIT 22-35  
 CM\_COLLAPSETREE 22-36  
 CM\_ERASERECORD 22-37  
 CM\_EXPANDTREE 22-38  
 CM\_FILTER 22-39  
 CM\_FREEDetailFIELDINFO 22-40  
 CM\_FREERECORD 22-42  
 CM\_HORZSCROLLSPLITWINDOW 22-43  
 CM\_INSERTDETAILFIELDINFO 22-44  
 CM\_INSERTRECORD 22-45  
 CM\_INSERTRECORDARRAY 22-47  
 CM\_INVALIDATEDetailFIELDINFO 22-49  
 CM\_INVALIDATERECORD 22-50  
 CM\_MOVETREE 22-52  
 CM\_OPENEDIT 22-53  
 CM\_PAINTBACKGROUND 22-55  
 CM\_QUERYCNRINFO 22-56  
 CM\_QUERYDETAILFIELDINFO 22-56  
 CM\_QUERYDRAGIMAGE 22-57  
 CM\_QUERYRECORD 22-59  
 CM\_QUERYRECORDEMPHASIS 22-60  
 CM\_QUERYRECORDFROMRECT 22-61  
 CM\_QUERYRECORDINFO 22-63  
 CM\_QUERYRECORDRECT 22-64  
 CM\_QUERYVIEWPORTRECT 22-65  
 CM\_REMOVEDetailFIELDINFO 22-66  
 CM\_REMOVERECORD 22-68  
 CM\_SCROLLWINDOW 22-70  
 CM\_SEARCHSTRING 22-70  
 CM\_SETCNRINFO 22-71  
 CM\_SETRECORDEMPHASIS 22-74  
 CM\_SETTEXTVISIBILITY 22-77  
 CM\_SORTRECORD 22-76  
 CMDSRC\_\* values 9-4, 10-37, 10-49, 10-91, 13-28  
 CN\_\* values 22-5  
     described 22-10  
 CN\_BEGINEDIT 22-10  
 CN\_COLLAPSETREE 22-10  
 CN\_CONTEXTMENU 22-12  
 CN\_DRAGAFter 22-12  
 CN\_DRAGLEAVE 22-15  
 CN\_DRAGOVER 22-16  
 CN\_DROP 22-18  
 CN\_DROPHELP 22-19  
 CN\_DROPNOTIFY 22-19  
 CN\_EMPHASIS 22-20  
 CN\_ENDEDIT 22-21  
 CN\_ENTER 22-22  
 CN\_EXPANDTREE 22-23  
 CN\_HELP 22-23  
 CN\_INITDRAG 22-24  
 CN\_KILLFOCUS 22-25  
 CN\_PICKUP 22-26  
 CN\_QUERYDELTA 22-27  
 CN\_REALLOCPSZ 22-28  
 CN\_SCROLL 22-29  
 CN\_SETFOCUS 22-29  
 CNRDRAINFO A-27, A-28  
 CNRDRAINIT A-32  
 CNRDRAWITEMINFO A-28  
 CNREDITDATA A-29  
 CNRINFO A-32, A-33  
 CNRLAZYDRAGINFO A-39  
 Code pages 32-1  
     ASCII 32-12  
     EBCDIC 32-21  
     Font support 32-4  
     OS/2 options for PM 32-3  
     OS/2 support for multiple 32-4  
 COL parameter A-52  
 COLOR A-40  
 color table F-1  
 combination-box control data 17-1  
 combination-box control window processing 17-1  
 container control window processing  
     data structures 22-4  
     icon size, how determined A-35  
     mini-icon size, how determined A-35  
     notification codes 22-10

- container control window processing (*continued*)
  - notification messages 22-5
  - purpose 22-1
  - styles and selection types 22-2
  - window messages 22-31
  - window words 22-2
- container views A-34
- contents and format of dialog template 31-24
- control classes 9-2
- control data 31-28
- control statements
  - predefined 31-30
- control window processing 9-2
- Control-Data A-44
- CONVCONTEXT A-40
- coordinates
  - dialog 31-24
- coordinates for dialogs 31-24
- COP parameter A-51
- CREATESTRUCT A-41
- CS\_\* values
  - window class styles 10-2
- CSBITMAPDATA A-42
- CSM\_QUERYINCREMENT 25-5
- CSM\_QUERYRADIUS 25-5
- CSM\_QUERYRANGE 25-6
- CSM\_QUERYVALUE 25-6
- CSM\_SETBITMAPDATA 25-7
- CSM\_SETINCREMENT 25-7
- CSM\_SETRANGE 25-8
- CSM\_SETVALUE 25-9
- CTIME A-44
- CURSORINFO A-43
- CV\_\* values
  - CNRINFO A-34
  - SEARCHSTRING A-185
  - view styles A-35
- CVR\_\* values 10-30

## D

- data
  - bit map D-1
- data area in a dialog template 31-27
- DC\_\* values A-61
- DDE\_\* values 29-1, 29-2, 29-3, A-47
- DDEINIT A-45
- DDESTRUCT A-46
- default colors 11-3, 12-3, 13-4, 14-1, 15-3, 17-2, 18-1, 20-2, 21-1, 25-2

- default dialog processing 10-103
- default message processing 10-1
- default window processing 9-2
- DEFAULTICON keyword 31-3
- DELETONOTIFY A-46
- DESKTOP A-48
- DEVOPENSTRUC A-49
- Dialog and Window Template statement 31-9
- dialog processing 10-103
  - default 10-103
  - language support 10-120
- dialog template
  - data-area information 31-27
  - format and contents 31-24
  - header information 31-26
  - item information 31-26
- direct manipulation messages 28-1
- directives 31-6
- DLGC\_\* values 10-105
- DLGTEMPLATE A-53
- DLGTEMPLATE statement 31-13
- DLGTITEM A-54
- DM\_DISCARDOBJECT 28-1
- DM\_DRAGERROR 28-1
- DM\_DRAGFILECOMPLETE 28-2
- DM\_DRAGLEAVE 28-4
- DM\_DRAGOVER 28-4
- DM\_DRAGOVERNOTIFY 28-7
- DM\_DROP 28-8
- DM\_DROPHELP 28-9
- DM\_DROPNOTIFY 28-10
- DM\_EMPHASIZETARGET 28-11
- DM\_ENDCONVERSATION 28-11
- DM\_FILERENDERED 28-12
- DM\_PRINTOBJECT 28-13
- DM\_RENDER 28-14
- DM\_RENDERCOMPLETE 28-15
- DM\_RENDERFILE 28-17
- DM\_RENDERPREPARE 28-18
- DMFL\_\* values. A-63
- DO\_\* values
  - DRAGINFO A-58
  - DRAGITEM A-61
- drag messages 28-1
- DRAGIMAGE A-56
- DRAGINFO A-57
- DRAGITEM A-58
- DRAGTRANSFER A-61
- DRF\_\* values A-60

DRG\_\* values A-56  
DRIVDATA A-63  
DRM\_\* values A-60  
DRT\_\* values A-59  
DT\_\* values 20-1  
dynamic data exchange messages 29-1

## E

EM\_CLEAR 12-6  
EM\_COPY 12-6  
EM\_CUT 12-7  
EM\_PASTE 12-8  
EM\_QUERYCHANGED 12-8  
EM\_QUERYFIRSTCHAR 12-9  
EM\_QUERYREADONLY 12-10  
EM\_QUERYSEL 12-11  
EM\_SETFIRSTCHAR 12-12  
EM\_SETINSERTMODE 12-13  
EM\_SETREADONLY 12-13  
EM\_SETSEL 12-14  
EM\_SETTEXTLIMIT 12-15  
EN\_\* values 12-4, 16-3  
entry field control data 12-3  
entry field control window processing 12-1  
ENTRYFDATA A-64  
ERRINFO A-65  
ERRORID A-65  
errors B-1, C-1  
ES\_\* dbcsvs 12-2  
ES\_\* values 12-1  
ESCMODE A-66  
ESCSETMODE A-67

## F

FACENAMEDESC A-68  
FATTR\_SEL\_\* values A-70  
FATTR\_TYPE\_\* values A-71  
FATTRS A-69  
FCF\_\* values 13-1  
FDM\_ERROR 10-106  
FDM\_FILTER 10-107  
FDM\_VALIDATE 10-108  
FDS\_\* values A-78  
FFDESCS A-71  
FI\_\* values 13-23  
FID\_\* values 13-1, 21-1  
FIELDINFO A-72

FIELDINFOINSERT A-75  
file dialog 10-106  
file format  
file formats  
    bit maps D-2  
    icon file D-2  
    pointer D-2  
FILEDLG A-77  
FIT parameter A-51  
FIXED A-81  
FNTF\_\* values A-84, A-85  
FNTM\_FACENAMECHANGED 10-111  
FNTM\_FILTERLIST 10-112  
FNTM\_POINTSIZEDCHANGED 10-113  
FNTM\_STYLECHANGED 10-114  
FNTM\_UPDATEPREVIEW 10-115  
FNFS\_\* values A-83  
font dialog 10-109  
FONTDLG A-82  
FONTMETRICS A-88  
fonts supplied with the OS/2 operating system E-1  
format and contents of dialog template 31-24  
frame control data 13-3  
frame control window processing 13-1  
FRAMECDATA A-98  
FS\_\* values 13-3

## G

general window styles 10-1  
GRADIENTL A-100

## H

HAB A-100  
HACCEL A-100  
HAPP A-101  
HATOMTBL A-101  
HBITMAP A-101  
HCAPS\_\* values A-103  
HCINFO A-102  
HDC A-102  
HDDF A-103  
header 31-26  
help manager messages 30-1  
HELINIT A-104  
HELPSUBTABLE A-106  
HELPTABLE A-108  
HENUM A-108

HEV A-108  
 HINI A-108  
 HLIB A-109  
 HM\_ACTIONBAR\_COMMAND 30-1  
 HM\_CONTROL 30-1  
 HM\_CREATE\_HELP\_TABLE 30-2  
 HM\_DISMISS\_WINDOW 30-3  
 HM\_DISPLAY\_HELP 30-4  
 HM\_ERROR 30-5  
 HM\_EXT\_HELP 30-7  
 HM\_EXT\_HELP\_UNDEFINED 30-8  
 HM\_GENERAL\_HELP 30-8  
 HM\_GENERAL\_HELP\_UNDEFINED 30-9  
 HM\_HELP\_CONTENTS 30-10  
 HM\_HELP\_INDEX 30-10  
 HM\_HELPSUBITEM\_NOT\_FOUND 30-11  
 HM\_INFORM 30-12  
 HM\_INVALIDATE\_DDF\_DATA 30-12  
 HM\_KEYS\_HELP 30-14  
 HM\_LOAD\_HELP\_TABLE 30-15  
 HM\_NOTIFY 30-15  
 HM\_QUERY 30-17  
 HM\_QUERY\_DDF\_DATA 30-19  
 HM\_QUERY\_KEYS\_HELP 30-20  
 HM\_REPLACE\_HELP\_FOR\_HELP 30-20  
 HM\_REPLACE\_USING\_HELP 30-21  
 HM\_SET\_ACTIVE\_WINDOW 30-22  
 HM\_SET\_COVERPAGE\_SIZE 30-23  
 HM\_SET\_HELP\_LIBRARY\_NAME 30-24  
 HM\_SET\_HELP\_WINDOW\_TITLE 30-25  
 HM\_SET\_OBJCOM\_WINDOW 30-25  
 HM\_SET\_SHOW\_PANEL\_ID 30-26  
 HM\_SET\_USERDATA 30-27  
 HM\_TUTORIAL 30-27  
 HM\_UPDATE\_OBJCOM\_WINDOW\_CHAIN 30-28  
 HMERR\_\* error constants 30-5  
 HMF A-109  
 HMODULE A-109  
 HMQ A-110  
 HMTX A-110  
 HMUX A-110  
 HOBJECT A-110  
 HPOINTER A-111  
 HPROGRAM A-111  
 HPS A-111  
 HRGN A-111  
 HSAVEWP A-112  
 HSEM A-112  
 HSPL A-112

HSTR A-112  
 HSWITCH A-113  
 HT\_\* values 10-50  
 HWND A-113

**I**  
 icon file format D-2  
 icon size, how determined A-35  
 ICONINFO A-113  
 IMAGEBUNDLE A-114  
 information tables  
     bit map D-1  
 initialization file G-1  
 interchange file format F-1  
 IPT A-115  
 items in a dialog template 31-26

**J**  
 JRN\_\* values 10-54

**K**  
 KC\_\* values 10-32  
 kerning A-98  
     enable A-71  
     number of pairs A-98  
 KERNINGPAIRS A-115  
 KERNINGPAIRS data structure A-115  
 keyboard control codes 10-32  
 keyboard resources 31-10  
 keyboard statements  
     keyboard 31-10

**L**  
 language support dialog processing 10-120  
 language support window processing 10-116  
 LBOXINFO A-115  
 LHANDLE A-117  
 LINEBUNDLE A-117  
 list box control data 14-1  
 list box control styles 14-1  
 list box control window processing 14-1  
 LIT\_\* values 14-8  
 LM\_DELETEALL 14-7  
 LM\_DELETEITEM 14-7  
 LM\_INSERTITEM 14-8

LM\_INSERTMULTITEMS 14-9  
 LM\_QUERYITEMCOUNT 14-10  
 LM\_QUERYITEMHANDLE 14-11  
 LM\_QUERYITEMTEXT 14-12  
 LM\_QUERYITEMTEXTLENGTH 14-13  
 LM\_QUERYSELECTION 14-13  
 LM\_QUERYTOPINDEX 14-14  
 LM\_SEARCHSTRING 14-15  
 LM\_SELECTITEM 14-16  
 LM\_SETITEMHANDLE 14-18  
 LM\_SETITEMHEIGHT 14-19  
 LM\_SETITEMTEXT 14-19  
 LM\_SETITEMWIDTH 14-20  
 LM\_SETTOPINDEX 14-21  
 LN\_\* values 14-3  
 LONG A-118  
 LS\_\* values 14-1  
 LSS\_\* values 14-15

## M

MAP parameter A-52  
 MARKERBUNDLE A-119  
 MATRIXLF A-120  
 MB2D A-121  
 MB2INFO A-121  
 menu control styles 15-1  
 menu control window processing 15-1  
 menu item attributes 15-3  
 menu item styles 15-2  
 MENU statement 31-9, 31-16  
 MENUITEM A-123  
 menus  
   pull-down 31-19  
   templates 31-20  
 message processing  
   introduction 9-1  
   notation conventions 9-4  
   types 9-1  
 message types 9-1  
 Metafile data format F-3  
 metafile restrictions F-1  
 metafiles  
   general rules F-1  
 MIA\_\* values 15-3  
 mini-icon size, how determined A-35  
 MINIRECORDCORE A-124  
 MIS\_\* values 15-2, 31-21  
 MIT\_\* values 15-12, 15-15, 15-23  
 MLE\_SEARCHDATA A-125  
 MLECTLDATA A-127  
 MLEMARGSTRUCT A-126  
 MLEOVERFLOW A-136  
 MLM\_CHARFROMLINE 16-9  
 MLM\_CLEAR 16-9  
 MLM\_COPY 16-10  
 MLM\_CUT 16-11  
 MLM\_DELETE 16-12  
 MLM\_DISABLEREFRESH 16-12  
 MLM\_ENABLEREFRESH 16-13  
 MLM\_EXPORT 16-14  
 MLM\_FORMAT 16-15  
 MLM\_IMPORT 16-16  
 MLM\_INSERT 16-17  
 MLM\_LINEFROMCHAR 16-18  
 MLM\_PASTE 16-18  
 MLM\_QUERYBACKCOLOR 16-19  
 MLM\_QUERYCHANGED 16-19  
 MLM\_QUERYFIRSTCHAR 16-20  
 MLM\_QUERYFONT 16-21  
 MLM\_QUERYFORMATLINELENGTH 16-21  
 MLM\_QUERYFORMATRECT 16-22  
 MLM\_QUERYFORMATTEXTLENGTH 16-23  
 MLM\_QUERYIMPORTEXPORT 16-24  
 MLM\_QUERYLINECOUNT 16-25  
 MLM\_QUERYLINELENGTH 16-25  
 MLM\_QUERYREADONLY 16-26  
 MLM\_QUERYSEL 16-27  
 MLM\_QUERYSELTEXT 16-29  
 MLM\_QUERYTABSTOP 16-29  
 MLM\_QUERYTEXTCOLOR 16-30  
 MLM\_QUERYTEXTLENGTH 16-31  
 MLM\_QUERYTEXTLIMIT 16-31  
 MLM\_QUERYUNDO 16-32  
 MLM\_QUERYWRAP 16-33  
 MLM\_RESETUNDO 16-33  
 MLM\_SEARCH 16-35  
 MLM\_SETBACKCOLOR 16-37  
 MLM\_SETCHANGED 16-37  
 MLM\_SETFIRSTCHAR 16-38  
 MLM\_SETFONT 16-39  
 MLM\_SETFORMATRECT 16-40  
 MLM\_SETIMPORTEXPORT 16-43  
 MLM\_SETREADONLY 16-43  
 MLM\_SETSEL 16-45  
 MLM\_SETTABSTOP 16-46  
 MLM\_SETTEXTCOLOR 16-46  
 MLM\_SETTEXTLIMIT 16-47

MLM\_SETWRAP 16-48  
 MLM\_UNDO 16-49  
 MLS\_\* values 16-2  
 MM\_DELETEITEM 15-10  
 MM\_ENDMENUMODE 15-10  
 MM\_INSERTITEM 15-11  
 MM\_ISITEMVALID 15-12  
 MM\_ITEMIDFROMPOSITION 15-13  
 MM\_ITEMPOSITIONFROMID 15-14  
 MM\_QUERYDEFAULTITEMID 15-15  
 MM\_QUERYITEM 15-15  
 MM\_QUERYITEMATTR 15-16  
 MM\_QUERYITEMCOUNT 15-18  
 MM\_QUERYITEMRECT 15-18  
 MM\_QUERYITEMTEXT 15-19  
 MM\_QUERYITEMTEXTLENGTH 15-20  
 MM\_QUERYSELITEMID 15-21  
 MM\_REMOVEITEM 15-22  
 MM\_SELECTITEM 15-23  
 MM\_SETDEFAULTITEMID 15-24  
 MM\_SETITEM 15-25  
 MM\_SETITEMATTR 15-26  
 MM\_SETITEMHANDLE 15-27  
 MM\_SETITEMTEXT 15-28  
 MM\_STARTMENUMODE 15-28  
 MPARAM A-129  
 MQINFO A-129  
 MRESULT A-130  
 MS\_\* values 10-8, 15-1  
 multi-line entry field control data 16-2  
 multi-line entry field control window processing 16-1  
 multiple-line statements 31-9  
     ACCELTABLE 31-10  
     ASSOCTABLE 31-12  
     DLGTEMPLATE 31-13  
     MENU 31-16  
     STRINGTABLE 31-22  
     WINDOWTEMPLATE 31-13

## N

notation conventions  
     messages 9-4  
 notebook control window processing  
     notification codes 23-3  
     notification messages 23-3  
     purpose 23-1  
     styles 23-1  
     window messages 23-7

NOTIFYDELTA A-130  
 NOTIFYRECORDEMPHASIS A-131  
 NOTIFYRECORDENTER A-132  
 NOTIFYSCROLL A-133

## O

OBJCLASS A-134  
 owner-notification messages 9-4  
 OWNERBACKGROUND A-135  
 OWNERITEM 10-110, A-136  
     owneritem parameter 22-7  
     WM\_DRAWITEM for container control 22-7  
     WM\_DRAWITEM for font dialog 10-110

## P

PAGEINFO A-138  
 PAGESELECTNOTIFY A-140  
 PANOSE A-140  
 PARAM A-144  
 parent/child/owner relationship 31-29  
 PC VKEY H-1  
 PCH A-147  
 PCSZ A-147  
 PDEVOPENDATA A-147  
 PFN A-148  
 PFNWP A-148  
 PID A-148  
 PIX A-148  
 PL\_ALTERED 10-5  
 PM\_\* names G-1  
 PM\_Q\_\* values A-50  
 pointer file format D-2  
 POINTERINFO A-169  
 POINTL A-170  
 POINTS A-171  
 PQMOPENDATA A-171  
 PRD\_\* values. A-150  
 PRDINFO3 A-149  
 PRDRIVINFO A-151  
 predefined control statements 31-30  
 predefined window classes 31-30  
 presentation parameters 31-28  
 presentation space  
     cached 13-15  
 PRESPARAMS A-151  
 PRFPROFILE A-154  
 PRINTDEST A-152

PRINTERINFO A-153  
 PRJ\_\* values. A-150, A-156  
 PRJ\_QS\_\* values. A-156  
 PRJINFO2 A-155  
 PRJINFO3 A-157  
 PROG\_\* values A-161  
 PROGCATEGORY A-160  
 PROGDETAILS A-160  
 PROGRAMENTRY A-159  
 PROGTYP E A-161  
 prompted entry field control window  
   processing 17-1  
 PRPORTINFO A-162  
 PRPORTINFO1 A-163  
 PRQINFO3 A-164  
 PRQINFO6 A-166  
 PRQPROCINFO A-168  
 PSZ A-171  
 pull-down menus 31-19  
 PVOID A-172  
 PWPOINT A-172

## Q

QFC\_\* values 13-21  
 QMSG A-172  
 QUERYRECFROMRECT A-173  
 QUERYRECORDRECT A-174  
 QWL\_USER in containers 22-2

## R

radio button 11-2  
 RECORDCORE A-175  
 RECORDINSERT A-177  
 RECTL A-178  
 RENDERFILE A-179  
 reserved messages 10-1  
 resource definitions 31-2  
 resource file specification 31-35  
 resource files  
   definitions 31-2  
   introduction 31-1  
   source file specification 31-35  
   syntax definitions 31-1  
 resource script file  
   keyboard resources specification 31-10  
   specification 31-2  
 resource script file specification  
   user-defined resources 31-4

resource statements  
   ACCELTABLE 31-10  
   ASSOCTABLE 31-12  
   dialog template 31-13  
   directives 31-6  
   DLGTEMPLATE 31-13  
   MENU item 31-18  
   MENU statement 31-16  
   multiple-line 31-9  
   single line 31-3  
   STRINGTABLE 31-22  
   user-defined 31-4  
   window template 31-13  
   WINDOWTEMPLATE 31-13  
 RGB A-180  
 RGB2 A-181  
 RGNRECT A-182  
 RT\_\* values 31-35

## S

SB\_\* values 10-51, 10-100, 27-3, 27-7  
 SBCDATA A-182  
 SBM\_QUERYPOS 18-4  
 SBM\_QUERYRANGE 18-4  
 SBM\_SETPOS 18-5  
 SBM\_SETSCROLLBAR 18-6  
 SBM\_SETTHUMB SIZE 18-7  
 SBS\_\* values 18-1  
 SC\_\* values 13-27  
 scroll bar control data 18-1  
 scroll bar control window processing 18-1  
 scroll bar styles 18-1  
 SDT\_\* values A-48  
 SEARCHSTRING A-183, A-184  
 SEGOFF A-185  
 SEPARATOR menu item 31-20  
 SFACTORS A-185  
 SHORT A-186  
 SIZEF A-186  
 SIZEL A-186  
 SLDCDATA A-187  
 slider control window processing  
   data structures 24-4  
   notification messages 24-5  
   purpose 24-1  
   styles 24-1  
   window messages 24-8  
 SLM\_ADDDETENT 24-8

- SLM\_QUERYDETENTPOS 24-8, 24-9
- SLM\_QUERYSCALETEXT 24-9, 24-10
- SLM\_QUERYSLIDERINFO 24-11
- SLM\_QUERYTICKPOS 24-13
- SLM\_QUERYTICKSIZE 24-14
- SLM\_REMOVEDETENT 24-15
- SLM\_SETSCALETEXT 24-16
- SLM\_SETSLIDERINFO 24-17
- SLM\_SETTICKSIZE 24-19
- SLS\_\* values 24-1
- SM\_QUERYHANDLE 20-4
- SM\_SETHANDLE 20-5
- SMHSTRUCT A-188
- source resource file 31-35
- SPBCDATA A-189
- SPBM\_OVERRIDESETLIMITS 19-4
- SPBM\_QUERYLIMITS 19-5
- SPBM\_QUERYVALUE 19-6
- SPBM\_SETARRAY 19-8
- SPBM\_SETCURRENTVALUE 19-9
- SPBM\_SETLIMITS 19-9
- SPBM\_SETMASTER 19-10
- SPBM\_SETTEXTLIMIT 19-11
- SPBM\_SPINDOWN 19-12
- SPBM\_SPINUP 19-13
- spin button control window processing 19-1
  - notification message 19-3
  - purpose 19-1
  - styles 19-1
- SPLERR A-190
- SS\_\* values 20-1
- standard bit-map formats D-1
- static control data 20-2
- static control styles 20-1
- static control window processing 20-1
- STR16 A-190
- STR32 A-190
- STR64 A-190
- STR8 A-191
- STRINGTABLE statement 31-9, 31-22
- STYLECHANGE A-191
- submenus 31-19
- SV\_\* values
  - effect on container icon size A-35
  - effect on container mini-icon size A-35
- SWBLOCK A-193
- SWCNTRL A-193
- SWENTRY A-194
- SWL\_\* values A-194

- SWP A-195
- SWP\_\* values 10-101, A-196

## T

- TBM\_QUERYHILITE 21-3
- TBM\_SETHILITE 21-4
- templates
  - dialog 31-24
  - format 31-21
  - menus 31-20
- TF\_\* values A-198
- TID A-197
- title bar
  - control data 21-1
  - control window processing 21-1
  - style 21-1
- TRACKINFO A-197
- TREEITEMDESC A-199
- TREEMOVE A-200
- triplets F-3

## U

- UCHAR A-201
- ULONG A-201
- user-defined resources 31-4
- USERBUTTON A-201
- USHORT A-202

## V

- value set control window processing
  - data structures 26-5
  - notification messages 26-6
  - purpose 26-1
  - styles 26-1
  - window messages 26-10
- VIA\_\* values
  - querying item attributes 26-12
  - setting item attributes 26-19
- VIOFONTCELLSIZE A-203
- VIOSIZECOUNT A-203
- virtual key definitions H-1
- VK\_\* values A-2
- VM\_QUERYITEM 26-10
- VM\_QUERYITEMATTR 26-11, 26-12
- VM\_QUERYMETRICS 26-14
- VM\_QUERYSELECTEDITEM 26-15



VM\_SELECTITEM 26-16  
VM\_SETITEM 26-17  
VM\_SETITEMATTR 26-19  
VM\_SETMETRICS 26-21  
VOID A-204  
VS\_\* values 26-1  
VSCDATA A-204  
VSDRAGINFO A-205  
VSDRAGINIT A-205  
VSTEXT A-206

## W

WC\_\* values 9-2, 21-1  
window class styles 10-2  
window processing  
    button control 11-1  
    combination-box control 17-1  
    container control 22-1  
    control 9-2  
    default 9-2, 10-1  
    entry field control 12-1  
    frame control 13-1  
    language support 10-116  
    list box control 14-1  
    menu control 15-1  
    multi-line entry field control 16-1  
    notebook control 23-1  
    prompted entry field control 17-1  
    scroll bar control 18-1  
    slider control 24-1  
    spin button control 19-1  
    static control 20-1  
    value set control 26-1  
WINDOWTEMPLATE statement 31-13  
WM\_ACTIVATE 10-5  
WM\_ACTIVATE (in Frame Controls) 13-8  
WM\_ACTIVATE (Language Support Dialog) 10-120  
WM\_ACTIVATE (Language Support Window) 10-116  
WM\_ADJUSTFRAMEPOS 13-8  
WM\_ADJUSTWINDOWPOS 10-7  
WM\_APPTERMINATENOTIFY 10-7  
WM\_BEGINDRAG 10-9  
WM\_BEGINSELECT 10-10  
WM\_BUTTON1CLICK 10-11  
WM\_BUTTON1DBLCLK 10-12  
WM\_BUTTON1DBLCLK (in Frame Controls) 13-9  
WM\_BUTTON1DBLCLK (in Multiline Entry Fields) 16-50  
WM\_BUTTON1DOWN 10-13  
WM\_BUTTON1DOWN (in Frame Controls) 13-10  
WM\_BUTTON1DOWN (in Multiline Entry Fields) 16-50  
WM\_BUTTON1MOTIONEND 10-14  
WM\_BUTTON1MOTIONSTART 10-15  
WM\_BUTTON1UP 10-16  
WM\_BUTTON1UP (in Frame Controls) 13-11  
WM\_BUTTON1UP (in Multiline Entry Fields) 16-51  
WM\_BUTTON2CLICK 10-17  
WM\_BUTTON2DBLCLK 10-18  
WM\_BUTTON2DBLCLK (in Frame Controls) 13-10  
WM\_BUTTON2DOWN 10-19  
WM\_BUTTON2DOWN (in Frame Controls) 13-10  
WM\_BUTTON2MOTIONEND 10-20  
WM\_BUTTON2MOTIONSTART 10-21  
WM\_BUTTON2UP 10-22  
WM\_BUTTON2UP (in Frame Controls) 13-11  
WM\_BUTTON3CLICK 10-23  
WM\_BUTTON3DBLCLK 10-23  
WM\_BUTTON3DOWN 10-24  
WM\_BUTTON3MOTIONEND 10-25  
WM\_BUTTON3MOTIONSTART 10-27  
WM\_BUTTON3UP 10-28  
WM\_CALCFRAMERECT 10-29  
WM\_CALCFRAMERECT (in Frame Controls) 13-12  
WM\_CALCVALIDRECTS 10-30  
WM\_CHAR 10-32  
    circular slider control 25-9  
    notebook control 23-30  
    slider control 24-20  
    value set control 26-22  
WM\_CHAR (Default Dialogs) 10-103  
WM\_CHAR (in Circular Slider Controls) 25-9  
WM\_CHAR (in Entry Fields) 12-16  
WM\_CHAR (in Frame Controls) 13-12  
WM\_CHAR (in List Boxes) 14-22  
WM\_CHAR (in Multiline Entry Fields) 16-52  
WM\_CHAR (in Notebook Controls) 23-30  
WM\_CHAR (in Slider Controls) 24-20  
WM\_CHAR (in Value Set Controls) 26-22  
WM\_CHORD 10-34  
WM\_CLOSE 10-35  
WM\_CLOSE (Default Dialogs) 10-104  
WM\_CLOSE (in Frame Controls) 13-13  
WM\_COMMAND 9-4, 10-37, 13-13  
WM\_COMMAND (Default Dialogs) 10-104  
WM\_COMMAND (in Button Controls) 11-5  
WM\_COMMAND (in Menu Controls) 15-5

WM\_CONTEXTMENU 10-38  
 WM\_CONTROL 9-4, 10-39  
     container control 22-5  
     notebook control 23-3  
     slider control 24-5  
     value set control 26-6  
 WM\_CONTROL (in Button Controls) 11-5  
 WM\_CONTROL (in Circular Slider Controls) 25-3  
 WM\_CONTROL (in Combination Boxes) 17-3  
 WM\_CONTROL (in Container Controls) 22-5  
 WM\_CONTROL (in Entry Fields) 12-4  
 WM\_CONTROL (in List Boxes) 14-3  
 WM\_CONTROL (in Multiline Entry Fields) 16-3  
 WM\_CONTROL (in Notebook Controls) 23-3  
 WM\_CONTROL (in Slider Controls) 24-5  
 WM\_CONTROL (in Spin Button Controls) 19-3  
 WM\_CONTROL (in Value Set Controls) 26-6  
 WM\_CONTROL (Language Support Dialog) 10-120  
 WM\_CONTROL (Language Support Window) 10-116  
 WM\_CONTROLPOINTER 10-40  
     circular slider control 25-4  
     container control 22-6  
     notebook control 23-4  
     slider control 24-6  
     value set control 26-7  
 WM\_CONTROLPOINTER (in Circular Slider Controls) 25-4  
 WM\_CONTROLPOINTER (in Container Controls) 22-6  
 WM\_CONTROLPOINTER (in Notebook Controls) 23-4  
 WM\_CONTROLPOINTER (in Slider Controls) 24-5  
 WM\_CONTROLPOINTER (in Value Set Controls) 26-7  
 WM\_CREATE 10-41  
 WM\_DDE\_ACK 29-1  
 WM\_DDE\_ADVISE 29-2  
 WM\_DDE\_DATA 29-3  
 WM\_DDE\_EXECUTE 29-4  
 WM\_DDE\_INITIATE 29-4  
 WM\_DDE\_INITIATEACK 29-6  
 WM\_DDE\_POKE 29-6  
 WM\_DDE\_REQUEST 29-7  
 WM\_DDE\_TERMINATE 29-8  
 WM\_DDE\_UNADVISE 29-9  
 WM\_DESTROY 10-42  
 WM\_DESTROYCLIPBOARD 27-1  
 WM\_DRAWCLIPBOARD 27-2  
 WM\_DRAWITEM 10-42  
     container control 22-7  
     font dialog 10-110  
     notebook control 23-4  
     slider control 24-6  
     value set control 26-8  
 WM\_DRAWITEM (in Container Controls) 22-7  
 WM\_DRAWITEM (in Font Dialog) 10-109  
 WM\_DRAWITEM (in Frame Controls) 13-13  
 WM\_DRAWITEM (in List Boxes) 14-4  
 WM\_DRAWITEM (in Menu Controls) 15-5  
 WM\_DRAWITEM (in Notebook Controls) 23-4  
 WM\_DRAWITEM (in Slider Controls) 24-6  
 WM\_DRAWITEM (in Value Set Controls) 26-8  
 WM\_ENABLE 10-43  
 WM\_ENABLE (in Button Controls) 11-14  
 WM\_ENABLE (in Multiline Entry Fields) 16-55  
 WM\_ENDDRAG 10-44  
 WM\_ENDSELECT 10-45  
 WM\_ERASEBACKGROUND 13-14  
 WM\_ERROR 10-46  
 WM\_FLASHWINDOW 13-15  
 WM\_FOCUSCHANGE 10-47  
 WM\_FOCUSCHANGE (in Frame Controls) 13-16  
 WM\_FORMATFRAME 10-48  
 WM\_FORMATFRAME (in Frame Controls) 13-16  
 WM\_HELP 9-4, 10-49  
 WM\_HELP (in Button Controls) 11-7  
 WM\_HELP (in Menu Controls) 15-6  
 WM\_HITTEST 10-50  
 WM\_HSCROLL 10-51  
 WM\_HSCROLL (in Horizontal Scroll Bars) 18-3  
 WM\_HSCROLLCLIPBOARD 27-2  
 WM\_INITDLG 10-52  
 WM\_INITDLG (Default Dialogs) 10-104  
 WM\_INITMENU 10-53  
 WM\_INITMENU (in Frame Controls) 13-17  
 WM\_INITMENU (in Menu Controls) 15-7  
 WM\_JOURNALNOTIFY 10-53  
 WM\_MATCHMNEMONIC 10-55  
 WM\_MATCHMNEMONIC (Default Dialogs) 10-105  
 WM\_MATCHMNEMONIC (in Button Controls) 11-14  
 WM\_MATCHMNEMONIC (in Static Controls) 20-5  
 WM\_MEASUREITEM 10-55  
 WM\_MEASUREITEM (in Frame Controls) 13-17  
 WM\_MEASUREITEM (in List Boxes) 14-5  
 WM\_MEASUREITEM (in Menu Controls) 15-7  
 WM\_MENUEND 10-56

WM\_MENUEND (in Menu Controls) 15-8  
 WM\_MENUSELECT 10-57  
 WM\_MENUSELECT (in Frame Controls) 13-18  
 WM\_MENUSELECT (in Menu Controls) 15-8  
 WM\_MINMAXFRAME 10-58  
 WM\_MINMAXFRAME (in Frame Controls) 13-5  
 WM\_MOUSEMAP 10-58  
 WM\_MOUSEMOVE 10-59  
 WM\_MOUSEMOVE (in Multiline Entry Fields) 16-55  
 WM\_MOVE 10-60  
 WM\_MSGBOXDISMISS 10-62  
 WM\_MSGBOXINIT 10-62  
 WM\_NEXTMENU 10-63  
 WM\_NEXTMENU (in Frame Controls) 13-18  
 WM\_NEXTMENU (in Menu Controls) 15-9  
 WM\_NULL 10-64  
 WM\_OPEN 10-65  
 WM\_OWNERPOSCHANGE 13-18  
 WM\_PACTIVATE 10-65  
 WM\_PAINT 10-66  
 WM\_PAINT (in Frame Controls) 13-19  
 WM\_PAINT (Language Support Window) 10-116  
 WM\_PAINT (Language Support Dialog) 10-120  
 WM\_PAINTCLIPBOARD 27-4  
 WM\_PCONTROL 10-67  
 WM\_PICKUP 22-78  
 WM\_PPAINT 10-68  
 WM\_PPAINT (Language Support Dialog) 10-121  
 WM\_PPAINT (Language Support Window) 10-117  
 WM\_PRESPARAMCHANGED 10-69  
     circular slider control 25-10  
     container control 22-79  
     notebook control 23-31  
     slider control 24-22  
     value set control 26-24  
 WM\_PRESPARAMCHANGED (in Circular Slider Controls) 25-10  
 WM\_PRESPARAMCHANGED (in Container Controls) 22-79  
 WM\_PRESPARAMCHANGED (in Notebook Controls) 23-31  
 WM\_PRESPARAMCHANGED (in Slider Controls) 24-22  
 WM\_PRESPARAMCHANGED (in Value Set Controls) 26-24  
 WM\_PSETFOCUS 10-69  
 WM\_PSIZE 10-70  
 WM\_PSYSCOLORCHANGE 10-71  
 WM\_QUERYACCELTABLE 10-72  
 WM\_QUERYBORDERSIZE 13-20  
 WM\_QUERYCONVERTPOS 10-72  
 WM\_QUERYCONVERTPOS (in Button Controls) 11-15  
 WM\_QUERYCONVERTPOS (in Entry Fields) 12-17  
 WM\_QUERYCONVERTPOS (in Frame Controls) 13-20  
 WM\_QUERYCONVERTPOS (in List Boxes) 14-23  
 WM\_QUERYCONVERTPOS (in Menu Controls) 15-29  
 WM\_QUERYCONVERTPOS (in Scroll Bars) 18-8  
 WM\_QUERYCONVERTPOS (in Static Controls) 20-6  
 WM\_QUERYCONVERTPOS (in Title Bar Controls) 21-4  
 WM\_QUERYDLGCODE 10-105  
 WM\_QUERYFOCUSCHAIN 13-21  
 WM\_QUERYFRAMECTLCOUNT 13-22  
 WM\_QUERYFRAMEINFO 13-23  
 WM\_QUERYHELPINFO 10-74  
 WM\_QUERYICON 13-23  
 WM\_QUERYTRACKINFO 10-74  
 WM\_QUERYWINDOWPARAMS 10-75  
     circular slider control 25-11  
     slider control 24-23  
     value set control 26-25  
 WM\_QUERYWINDOWPARAMS (in Button Controls) 11-15  
 WM\_QUERYWINDOWPARAMS (in Circular Slider Controls) 25-11  
 WM\_QUERYWINDOWPARAMS (in Entry Fields) 12-17  
 WM\_QUERYWINDOWPARAMS (in Frame Controls) 13-24  
 WM\_QUERYWINDOWPARAMS (in List Boxes) 14-23  
 WM\_QUERYWINDOWPARAMS (in Menu Controls) 15-30  
 WM\_QUERYWINDOWPARAMS (in Multiline Entry Fields) 16-56  
 WM\_QUERYWINDOWPARAMS (in Scroll Bars) 18-8  
 WM\_QUERYWINDOWPARAMS (in Slider Controls) 24-23  
 WM\_QUERYWINDOWPARAMS (in Static Controls) 20-6  
 WM\_QUERYWINDOWPARAMS (in Title Bars) 21-5  
 WM\_QUERYWINDOWPARAMS (in Value Set Controls) 26-25

WM\_QUIT 10-76  
 WM\_REALIZEPALETTE 10-78  
 WM\_RENDERALLFMTS 27-4  
 WM\_RENDERFMT 27-5  
 WM\_SAVEAPPLICATION 10-78  
 WM\_SEM1 10-79  
 WM\_SEM2 10-80  
 WM\_SEM3 10-81  
 WM\_SEM4 10-82  
 WM\_SETACCELTABLE 10-83  
 WM\_SETBORDERSIZE 13-25  
 WM\_SETFOCUS 10-83  
 WM\_SETFOCUS (Language Support Dialog) 10-122  
 WM\_SETFOCUS (Language Support Window) 10-118  
 WM\_SETHelpInfo 10-84  
 WM\_SETICON 13-25  
 WM\_SETSELECTION 10-85  
 WM\_SETWINDOWPARAMS 10-86  
     circular slider control 25-12  
     slider control 24-24  
     value set control 26-26  
 WM\_SETWINDOWPARAMS (in Button Controls) 11-15  
 WM\_SETWINDOWPARAMS (in Circular Slider Controls) 25-12  
 WM\_SETWINDOWPARAMS (in Entry Fields) 12-18  
 WM\_SETWINDOWPARAMS (in Frame Controls) 13-26  
 WM\_SETWINDOWPARAMS (in List Boxes) 14-23  
 WM\_SETWINDOWPARAMS (in Menu Controls) 15-30  
 WM\_SETWINDOWPARAMS (in Multiline Entry Fields) 16-57  
 WM\_SETWINDOWPARAMS (in Scroll Bars) 18-9  
 WM\_SETWINDOWPARAMS (in Slider Controls) 24-24  
 WM\_SETWINDOWPARAMS (in Static Controls) 20-7  
 WM\_SETWINDOWPARAMS (in Title Bar Controls) 21-5  
 WM\_SETWINDOWPARAMS (in Value Set Controls) 26-26  
 WM\_SHOW 10-87  
 WM\_SINGLESELECT 10-87  
 WM\_SIZE 10-88  
     notebook control 23-32  
     value set control 26-26  
 WM\_SIZE (in Frame Controls) 13-26  
 WM\_SIZE (in Notebook Controls) 23-32  
 WM\_SIZE (in Value Set Controls) 26-26  
 WM\_SIZE (Language Support Dialog) 10-122  
 WM\_SIZE (Language Support Window) 10-118  
 WM\_SIZECLIPBOARD 27-6  
 WM\_SUBSTITUTESTRING 10-89  
 WM\_SYSCOLORCHANGE 10-90  
 WM\_SYSCOLORCHANGE (Language Support Dialog) 10-122  
 WM\_SYSCOLORCHANGE (Language Support Window) 10-119  
 WM\_SYSCOMMAND 10-91, 11-7, 13-27, 15-31  
 WM\_SYSCOMMAND (in Title Bar Controls) 21-2  
 WM\_SYSVALUECHANGED 10-92  
 WM\_TEXTEDIT 10-93  
 WM\_TIMER 10-94  
 WM\_TRACKFRAME 10-95  
 WM\_TRACKFRAME (in Frame Controls) 13-29  
 WM\_TRACKFRAME (in Title Bar Controls) 21-2  
 WM\_TRANSLATEACCEL 10-95  
 WM\_TRANSLATEACCEL (in Frame Controls) 13-30  
 WM\_TRANSLATEMNEMONIC 10-96  
 WM\_TRANSLATEMNEMONIC (in Frame Controls) 13-30  
 WM\_UPDATEFRAME 10-97  
 WM\_UPDATEFRAME (in Frame Controls) 13-30  
 WM\_VRNDISABLED 10-98  
 WM\_VRNENABLED 10-98  
 WM\_VSCROLL 10-99  
 WM\_VSCROLL (in Vertical Scroll Bars) 18-3  
 WM\_VSCROLLCLIPBOARD 27-7  
 WM\_WINDOWPOSCHANGED 10-101  
 WNDPARAMS A-207  
 WPOINT A-208  
 WRECT A-208  
 WS\_\* values 10-3

## X

XFM parameter A-51  
 XLT parameter A-52  
 XYWINSIZE A-208




© IBM and OS 2 are registered trademarks of the  
International Business Machines Corporation



© IBM Corp. 1994

All Rights Reserved

**25H7191**  
**G25H-7191-00**

 Printed on recycled paper

Printed in U.S.A.



G25H-7191-00



P25H7191