Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

--------------------------------------------

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM
Micro Channel
OS/2
Operating System/2
Presentation Manager
WIN-OS/2
Workplace Shell
XGA

The following terms are trademarks of other companies:
C-bus is a trademark of Corollary, Inc.
PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.
UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.
Windows is a trademark of Microsoft Corporation.
MicroFocus Cobol MicroFocus Corporation.

Other trademarks are trademarks of their respective companies.

--------------------------------------------

Summary of Amendments

This is in interim update to the current edition of this book published by the ITSO. It is available in on-line form only.

This update contains corrections and additions to the original edition.

The following is a summary of the amendments included in this edition. Volume and Page number references refer to the original printed edition (SG24-4640-00).

Interim update 0.1

Vol1 Page 31, 1.10 Exceptions
Change heading of exception table to say # instead of TRAP

Vol1 Page 196, The Kernel Semaphore (KSEM)
Add an additional comment about Event KSEMs.

Vol1 Page 338, Vol2 Page 274, Glossary
Re-worded glossary entry for BMP.

Vol1 Page 350, Vol2 Page 287, Glossary
Added references to Vol 4 in VMKH entry.

Vol1 Page 350, Vol2 Page 287, Glossary
Re-worded glossary entry for UVI RT.

Vol2 Page 53, 2.2 Dump Decompression
DCOMP added to DF user guide.

Vol2 Page 170, 3.4.5.6 Physical Device Driver Header
Corrected .D DEV note about DevInt

Vol2 Page 198, MA - Format Memory Arena Records
Added description of hashing table.

Vol2 Page 227, 3.4.17.1 Idle PFs and 3.4.17.2 Free PFs
Clarify description of how to locate idle and free PFs.

Vol2 Page 251, 3.4.21 PB - Display Blocked Thread Information
Add an additional comment about Event KSEMs.

Vol2 Page 261, 3.4.24 Display User's Registers
.R command EI flag should read DI flag when bit 9 is 0 for Flag register bit mnemonics.

Vol2 Page 266, 3.4.26 .R - Set or Display Default Thread Slot
.S, command S option edited.

Vol4 Pages 27 and 32, 2.8.4 LogGetEntries
Removed references to LogGetEntries. This API is not supported.

Vol4 Page 49, 3.2.2 System Anchor Segment (SAS)
Correct offsets in SAS base section and inserted length field for SAS_Info_Data.

Vol4 Page 52 VMBH BMP Header Structure
Structure added to miscellaneous section

Vol4 Pages 80, 82, 83, and 88
Amended VMAR, VMAL, PF and VP structures to show correct layout of bit fields. Added pf_block to PF structure.

Vol4 Pages 88
Kernel Heap Structures added.

Vol4 Page 184, 3.7.3 System File Table Entry for OS/2 Warp 3.0
Corrected SFT length of sft_sfi and offsets of fields following.

Vol4 Page 210, 3.8.3 PDD IRQ Information Blocks
Corrected title of PDD IQR to PDD IRQ

Vol4 Page 245, 4.4 Standard GDT Assignments
Deleted note at end of table. Added new note before table.

Vol4 Page 280, 4.9 OS/2 FixPak to Build Level Cross-Reference.
Corrected build levels for Warp GA, Connect and title of 8.200

Vol4 Page 210, 3.8.3 PDD IRQ Information Blocks
Corrected VMOwner info for DIRQ and IRQI.

*Interim update 0.2*

Vol1 Page 101, Predefined Dynamic Trace Events
MONCALLS added

Vol1 Page 103, TRCUST, The Dynamic Trace Customizer
Change title

Vol1 page 130, DosEnterCriticalSection ....
'crt' state is incorrectly described.

Vol1 page 134, The Dispatcher, Priorities and Dispatching Classes
 Added note at end of section on running in the kernel and device drivers.

Vol1 Page 135, The Status of a Thread
'crt' state is corrected.

Vol1 Page 137, A Form for Unwinding Stacks
Removed from INF version.

Vol1 page 202, Involuntary Suspension
Critical Section: typo, helped should read held.

Vol1 page 202, Involuntary Suspension
Pre-emption: added note about kernel mode.

Vol1 page 308, How to find the MQ of any Thread
AAB at TLMA offset +0x8 (not +0xc)

Vol1 page 308, How to find the MQ of any Thread
Expanded note at the bottom of the page.

Vol1 page 317, Finding the System Queue
Current read pointer should be offset +0x0c (not +0x0e).

Vol2 Page 126, Q Command
Typos: cases->causes, memu->menu

Vol2 Page 129, 3.4.24 Display User's Registers
.R command NV flag value should be 0.

Vol2 Page 255, .PQ Scheduler Prioity Queues
.ptobPqQRunner should say 'run' state

Vol2 Page 261, 3.4.24 Display User's Registers
.R command NV flag value should be 0.

Vol3 Page 45, Major Code Assignments
MONCALLS separately listed

Vol3 Page 218-245, PMSHAPI
Obsolete tracepoints removed.

Vol4 Page 188, Record Lock Record for OS/2 Warp V3.0
GDT_FSC should read GDT_RLR

Vol4 Page 88, Virtual Page Structure
.vpf_s type should be S and vp_flink type should be D.

*Interim update 0.3*

Vol1 Pages 245 14.1.2.1 Who Owns Virtual Memory and Who Allocated it?
Correct typographic error: pseud-objects -> pseudo-object (on-line version only)
Vol2 Page 80, List of Internal Commands
.L should read L (printed edition only)

Vol2 Page 52 Dump Formatter Installation
In printed copy only, Note: DF&US.RET.EXE and DF&US.DEB.EXE should read df_ret.exe and df_deb.exe

Vol2 Page 214 3.4.16.2 Pseudo-Object Records
Correct typographic error in Notes: (psuedo)

Vol2 Pages 226 3.4.17.1 Free Page Frame Structures
Correct typographic error in Notes: (psuedo)

Vol2 Pages 227 3.4.17.2 Idle Page Frame Structures
Correct typographic error in Notes: (psuedo)

Vol3 Page 17 RETEP
Note added about modern C compilers.

Vol3 Page 5
Note added following first paragraph.

Vol3 Page 45 3.4 Trace Major and Minor Code Assignments
Table extended to include major codes reserved for other components.

Vol4 Pages 9 and 10, 2.1.2 DevHlp_SysRAS
References to DevHlp_SysTrace and DevHlp_SysRAS should read DevHlp_RAS.

Vol4 Page 14 2.2.1 Trace Buffer Structures
CHECK KEY: filed->field, Exclusively starts with Upper case E.

Interim update 0.4

Vol1 Page xx Acknowledgments
Add Joanna Hodgson.

Vol1 Page 9 1.3.5.2 Descriptor Flags
Correct definitions of bit 53 and 54

Vol1 Page 10 1.3.5.3 Descriptor Table Summary
Correct VDM considerations for IDT and LDT

Vol1 Page 196 The Kernel Semaphore
Clarify use of KSEM blockids

Vol1 Page 295 14.1.3.5
Added more useful PM message queue and window symbols

Vol1 Page 342 Interrupt Descriptor Table
Added information on VDM use of IDTs.

Vol1 Page 343 Local Descriptor Table
Added information on VDM use of LDTs.

Vol2 Page 14 Forcing a System Dump from the Kernel Debugger
Add .SYSDUMP command to first paragraph.

Vol2 Page 17-18 Forcing a System Dump from the Kernel Debugger
Add RegSA for Pentium processor support.

Vol2 Page 41 1.5 Kernel Debugger Breakpoints
Update XCPTBuildR3DispatcherStack
Add DOS32R3EXCEPTIONDISPATCHER

Vol2 Page 45-46 1.6 Exception Logic
Correct information about local exception handlers used by the system and the exception handling logic. Add details of Dos32R3ExceptionDispatcher and Dos32ExceptionCallBack.

Vol2 Page 48 1.6.2 Exception Handling - Overview diagram
DosRaiseException flow corrected. Dos32ExceptionDispatcher

Vol2 Page 86-87 3.3.6 BL command
Update for I/O breakpoints.

Vol2 Page 89 3.3.8 BR
Add I/O breakpoint.

Vol2 Page 99 3.3.17.1 Descriptor formats Table 4
Clarify BIG C32 definitions

Vol2 Page 105 3.3.20.1 DP command
Add note about invalidity of PDEs under DF.

Vol2 Page 127 3.3.35.1 R command
Add note about addressing mode.

Vol2 Page 127 - 129 3.3.35.1 R command
Remove reference to 24-bit registers and move GDTB and IDTB to 32-bit registers.

Vol2 Page 128 3.3.35.1 R command
Add CR4 to 32-bit reg syntax diagram

Vol2 Page 131 3.3.35.1 R command
Add note on CR4 and correct 32-bit register

Vol2 Page 137 3.3.38 U command
Add note about V8086 mode addressing.

Vol2 Page 147 External Commands
.O command added.

Vol2 Page 147 External Commands
.SYSDUMP command added.

Vol2 Page 148 3.4.1 .? command
Updated note.
Vol2 Page 161 3.4.5.11 .D
typo in warning formate->format
Vol2 Page 161 3.4.5 .D
Note added about <512 byte segments
Vol2 Page 167 3.4.5.5 figure 33
Add remark about fsctr
Vol2 Page 168 3.4.5.5 figure 33
Add note about KSEM blockids
Vol2 Page 174 3.4.5.8 .D MFT
Note added about ALLSTRICT kernel
Vol2 Page 196 .M
Added note about defaults for FP29 and V4
Vol2 Page 182 3.4.5.11 .D SEM32 etc..
pName typo corrected no->not
Vol2 Page 190 3.4.1 .lm command
Added I option. Updated Note. Update Results and Notes.
Vol2 Page 204 .MC
Added DF icon
Vol2 Page 205 .MC
Added additional note about early versions of DF
Vol2 Page 210 .ML Results and Notes
Updated warning about freed memory.
Vol2 Page 213 .MO Results and Notes
Updated note about the need for kernel symbols
Vol2 Page 225 .MP Syntax
Added I option and clarified syntax.
Vol2 Page 226 .MP R and L option Warnings.
Updated warnings for use of R and L
Updated warnings for use of R and L
Vol2 Page 237 .O command
.O command added.
Vol2 Page 247 .PB Sta
Note about fix
Vol2 Page 250 .PB Notes:
Updated note on ChildWait.
Vol2 Page 258 3.3.35.1 .R command
Add note about addressing mode.
Vol2 Page 266 .SYSDUMP
.SYSDUMP command added.
Vol4 Page 41 and 89, 3.0 and 3.5 Scheduler control blocks
ljmp structure added to list of scheduler structures.
Vol4 Page 96 3.5.1.7 Exception Handling - Overview diagram
DosRaiseException flow corrected. Dos32ExceptionDispatcher replaced _xcptExceptionDispatcher.
Vol4 Page 125 long-jump buffer
ljmp structure added

Interim update 0.5

Introduction to the current edition
Updated and fixed typos.
Vol1 Page 103 Chapter 8 TRCUST The Dynamic Trace Customizer
Remove duplicate chapter from volume 1
Vol1 Page 81 5.6.1.1 How to find the TSS
Add additional fields to the TSS
Vol1 Page 129 11.1.2 Multiprocessor Methods - Spin Locks
Correct formatting error in note about LOCK prefix.
Vol1 Page 142, 13.1.1.1 Address Space Arenas and Regions
Add note about Protected Region post FP19 Warp 3.0
Vol1 Page 143, 13.1.1.1 Address Space Arenas and Regions
Add note about Packed Region post FP19 Warp 3.0
Vol1 Page 255,256 Exploring Memory Management, Private Arena private data
Vol1 Page 260, Exploring Memory Management, Finding Who Owns Memory
Note about .mam after 3.0 FP29 and 4.0 GA
Vol1 Page 322, 14.1.4.1 Ring 0 Loop Dump Analysis Example
Add note about fsd and dd system owners after 3.0 FP29 and 4.0 GA
Vol2 Page 1, Kernel Debugger User Guide
Add up to date URL and FTP information.
Vol2 Page 46, 1.6 Trap and Exceptions Processing
Added information about VSU and clarified DelayHardError.
Interim update 0.6
Interim update 0.7

Vol1 page 127 10.2 Steps to Diagnose a Loop
  Typo in online version: threas-> threads
Vol1 page 129 11.1.2 Multiprocessor Methods - Spin Locks
  Typo in online version: Exchange -> Exchange
Vol1 page 130 11.1.3 DosEnterCriticalSection and DosExitCriticalSection
  Reword second sentence for better understanding.
Vol1 page 147 13.1.1.2 Virtual Address Space Management
  Add _ahvmShr and correct label typos for __ahvmShr and __ahvmsys
Vol2 page 45-46 1.6 Exception Logic
  Add parameter information for _XCPTBuildR3DispatcherStack.
Vol2 page 190s and 192 3.4.10 .LM - Format Loader Structures
  Add references to RASKDATA.
Vol3 page 8 2.1.3.1 Source Level Symbolic Support
  Updated information on supported compilers and added reference to VisuAge and DEDEL.
Vol3 page 9 2.1.3.2 MAP File Support
  Added reference to /l
Vol3 page 9 2.1.3.3 Building a Module
  Added reference to DEBDEL
Vol3 page 11 2.1.2 TDF and TFF File Usage
  Added references to latest tools (DTRACE, TRACEGET, TRSPOOL, TRACE /Q)
Vol3 page 13 2.2.2 TSF Header
  Added TDFID and changed MAJOR to be optional.
Vol3 page 13 2.2.2 TSF Header
  Updated major code range for RAS Enhancements
Vol3 page 14 2.2.3 Typelist Definition
  Clarified use of TYPELIST
Vol3 page 14 2.2.4 Grouplist Definition
  Clarified use of GROUPLIST
Vol3 page 15 2.2.5 Tracepoint Definition
Added new RETEP options to syntax.

Vol3 page 16 2.2.5.1 MINOR Keyword
Added reference to /D

Vol3 page 16 2.2.5.2 TP Keyword
Clarified use of @STATIC. Referenced MAKETSF. Referenced /PREINV. Updated RETEP description for new sub-keywords. Corrected invalid opcode list.

Vol3 pages 20-22 2.2.5.7 FMT Keyword
Clarified use of %P, %C, %U. Added note on TRACEFMT dump format. Clarified not for CMVC users.

Vol3 page 28 2.2.5.15 Address Specification
Added note about /l and name length limitations.

Vol3 page 28 2.2.5.15 Address Specification - Flat Register Form
Updated 32-bit register lists.

Vol3 page 29 2.2.5.15 Address Specification - Segmented Register Form
Updated 16-bit register lists. Clarify use or R prefix.

Vol3 page 31 2.3 Formatting Trace Data
Clarify use of %P and %R. Add details of prefix format.

Vol3 pages 38 2.5.1 External Messages
Remove this section. These messages are not generated.

Vol3 pages 39-42 2.5.2 Internal Messages
Remove this section. These messages are not generated.

Vol3 page 44 3.2 Group Qualifiers
Added new groups.

Vol3 page 44 3.3a DosXxxx API Pre-invocation Tracepoints
Added section describing return addresses.

Vol3 page 45 3.4 Major Code Cross-Reference
Added new assignments.

Vol3 page 55 3.4.7 Kernel Services Trace Events
Add indirected APIs

Vol3 page 96 3.4.7 OS2KRNLD Trace Events
Remove this section since identical with OS2KRNLR

Vol3 page 164 3.4.21 DOSCALL1.DLL Trace Events
Add indirected APIs

Vol3 page 168 3.4.21 DOSCALL1.DLL Trace Events
Add parameters to Dos32R3ExceptionDispatcher, Dos32ExceptionCallBack, xcptExecuteExceptionHandler, UT16_RETURN and UT32_RETURN tracepoints.

Vol3 page 209 3.4.30 QUECALLS.DLL Trace Events
Add indirected APIs

Vol3 page 209 3.4.33 SESMGR.DLL Trace Events
Add indirected APIs

Vol3 page 203 3.4.28 OS2CHAR.DLL Trace Events
Added CharBuffer to KbdStringIn Post-Invocation.

Vol3 page 345 3.4.51 PMGPIR Trace Events
Remove this section since identical with PMGPI

Vol3 page 345 3.4.51 PMGPIR Trace Events
Remove this section since identical with PMGPI

Vol3 page 382 3.4.54 PMGPID Trace Events
Remove this section since identical with PMGPI

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands
Change title to CONFIG.SYS RAS Statements

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands
Added SCKILLFEATUREENABLED.

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands
Added RASKDATA

Vol4 page 2 1.2a RASKDATA
Inserted RASKDATA section.
Preface

Debugging problems is essentially an iterative process of hypothesis, test and conclusion that aims to eliminate the irrelevant and therefore focus on the probable causal area.

To engage this process successfully one needs to be equipped with an innate ability to think laterally coupled with sufficient knowledge of the environment in which the problem persists and above all else to be able to use the tools that extract information from the system under diagnosis.

This scenario applies as much to first level problem determination (PD) as it does to the software developer who is engaged in detailed analysis of his programs' behaviour.

Information and tools to aid first level problem determination is relatively accessible. Technical literature is available from IBM and books stores that will fulfil the needs of the first level PD analyst. For example, the reader is invited to consult the following IBM Red-book publications to achieve an all-round high-level technical appreciation of the OS/2 environment:

- The Technical Compendium Volume 1 - Control Program
- The Technical Compendium Volume 2 - DOS and Windows Environment
- The Technical Compendium Volume 3 - Presentation Manager and Workplace Shell
- The Technical Compendium Volume 4 - Application Development
- The Technical Compendium Volume 5 - The Print Sub-system

The problem analysis level that is less well provided for is that which involves internal knowledge of the OS/2 operating system and its diagnostic tools. This is the level at which Service personnel, System Programmers and Software Developers work. It is this audience to which the OS/2 Debugging Handbooks are directed.

An inevitable consequence of working at a deep technical level is that the amount of information one could amass is vast. Given time constraints and the need to publish useable material before it became obsolete we had to make certain compromises for the first edition. The following principles guided us in making decisions about which material to include:

- Material that is adequately documented elsewhere is referenced but not included.
- Accurate reference documentation for the diagnostic tools and facilities available for OS/2 has been given priority over worked examples and OS/2 Internals reference material.
Internals information has centred around the base operating system - that is, the kernel.

We hope to remedy some of these short-comings in future revisions of this book and in companion volumes. Updates to the on-line version of this book will be made available via the Developer Connection CDROM.

The current printed edition contains full reference material for the following OS/2 System diagnostic facilities:

- System Trace
- System Dump
- Kernel Debugger

In addition to these topics we have included an introductory guide to problem determination. This provides a resumé of the hardware and software environment and an introduction to using the dump formatter and kernel debugger.

Throughout this book we assume the availability of and familiarity with two co-requisite publications:


This should be consulted as the authoritative source for hardware architectural information.

The Design of OS/2 by H.M. Deitel and M.S. Kogan.

This should be consulted for an overview of the internal operation and architecture of OS/2.

This book is supplied with a CDROM whose contents are:

- Sample exercises to accompany Volume 1, Introduction to Debugging. These take the form of system dumps of typical problems in application programs.

- On-line version of this book. This is slightly more up-to-date than the printed version and includes more worked examples. This is an .INF file and should be viewed using the OS/2 VIEW.EXE program. Much use has been made of hypertext links, which direct the user to the glossary. From the glossary it is possible to link to related material in other sections of the book.

- The OS/2 Problem Determination Package (OS2PDP), which includes the dump formatter, symbol files, and trace customiser (TRCUST).

Unless otherwise stated the material in this book may be assumed to be applicable to OS/2 Warp version 3.0 (ALLSTRICT Kernel).

As indicated above, work on this subject matter can never be complete. We intend build on and update the material in this edition. In order to address the areas in most need of attention we invite the reader to fill in the Reader’s Comment Form with their suggestions.

--------------------------------------------

Related Publications

Throughout this book we assume the availability and familiarity with two co-requisite publications:


The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.


The OS/2 Technical Library Control Program Programming Reference Version 2.00., S10G-6263-00.

The Design Workbook for Cruiser (OS/2 2.0) Volumes 1 - 13. IBM Internal Publication

The Final Programming Functional Specifications for Cruiser (OS/2.0). IBM Internal Publication.
Installable File Systems for OS/2 Version 2.0. IBM OEMI Publication.
The OS/2 1.0 System Trace Facility. IBM OEMI publication.
The PS/2 2.0 Error Logging Functions. IBM OEMI publication.
OS/2 2.0 Proc Lang 2/REXX Reference, S10G-6268-00
OS/2 2.0 Proc Lang 2/REXX User Guide, S10G-6269-00
OS/2 WARP Control Program Programming Guide, G25H-7101-00
OS/2 WARP Control Program Programming Reference, G25H-7102-00
OS/2 WARP PM Basic Programming Guide, G25H-7103-00
OS/2 WARP PM Advanced Programming Guide, G25H-7104-00
OS/2 WARP GPI Programming Guide, G25H-7106-00
OS/2 WARP GPI Programming Reference, G25H-7107-00
OS/2 WARP Workplace Shell Programming Guide, G25H-7108-00
OS/2 WARP Workplace Shell Programming Reference, G25H-7109-00
OS/2 WARP IPF Programming Guide, G25H-7110-00
OS/2 WARP Tools Reference, G25H-7111-00
OS/2 WARP Multimedia Subsystem Programming, G25H-7113-00
OS/2 WARP Multimedia Programming Reference, G25H-7114-00
OS/2 WARP PM Programming Reference Volume I, G25H-7190-00
OS/2 WARP PM Programming Reference Volume II, G25H-7191-00
Technical Reference - Personal Computer AT, Part Number 1502494
Personal System/2 and Personal Computer BIOS Interface Technical Reference - Part Number 68X2341

--------------------------------------------
International Technical Support Organization Publications

OS/2 Warp Connect, GG24-4505
OS/2 Warp Generation, Vol.1, SG24-4552
OS/2 Warp Version 3 and BonusPak, GG24-4426
Multimedia in Warp, GG24-2516
The Technical Compendium Volume 1 - Control Program, GG24-3730
The Technical Compendium Volume 2 - DOS and Windows Environment, GG24-3731
The Technical Compendium Volume 3 - Presentation Manager and Workplace Shell, GG24-3732
The Technical Compendium Volume 4 - Application Development, GG24-3774
The Technical Compendium Volume 5 - The Print Sub-system, GG24-3775
A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOCAT TXT. This package is updated monthly.

How to Order ITSO Redbooks
IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-445-9269. Most major credit cards are accepted. Outside the USA, customers should contact their local IBM office. Guidance may be obtained by sending a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

--------------------------------------------

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following URL:

http://www.redbooks.ibm.com/redbooks

IBM employees may access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks home page:


--------------------------------------------

Acknowledgements

The authors of this book are:
Pete Guy
IBM SDO, Austin

Richard Moore
IBM PSP EMEA

Redbook project developed by:

Tim Sennitt
ITSO Boca Raton, Center

This book could not have reached publication without the encouragement, help and support from a number of colleagues and friends. In particular we would like to thank the following:

Tim Sennitt for his help in preparing the printed material and doing much of the donkey-work to bring this to publication.

Joanne Rearnkham, Barry Bryan and David Jaramillo for their support in enabling access to the materials necessary to produce this book.
Chris Perritt and Glen Brew for making available the original Design Workbook and Functional Specifications for OS/2 2.0.

Charlie Schmitt for his original work on converting the kernel debugger code into a dump formatter.

Jeff Mielke and David Jaramillo for their work on PMDF, the structure compiler and continued work on the dump formatter.

Allen Gilbert for making available documentation on System Trace, which has been reproduced in an edited form in this book. Also, for making available an early version of the dump formatter without which it would not have been possible to develop the original Dump Formatter class.

Doug Azzarito for supplying the material on Kernel Debugger Remote Debug Set-up.

James Taylor for providing the basis of the lab exercises relating to PM hangs.

Marie Jazynka, one of the first OS/2 debuggers, for patient encouragement of a great many OS/2 debugging people.

Joanna Hodgson for preparing some of the Warp 3.0 fix pack 29 and Warp 4.0 updates.

Our management teams, without whose foresight and support none of this work would ever have started. These include:

- Hermann Lamberti General Manager for PSM EMEA; Gordon Bell - director PSM EMEA Technical Marketing; Chris Brown - manager PSM OEM and Enterprise Technical Marketing and Brian Rose - manager PSM Project Office; Roy Aho - Director of the Solution Developer Technical Support Center, for encouraging the beginnings of this several years ago; Terry Gray, manager of Platform Competency and Operation, within Solution Developer Technical Support, Austin.

Finally to Sarah-Jane for supporting many very extended working days and weeks.

-------------------------------

ITSO Technical Bulletin Evaluation

The OS/2 Debugging Handbook Library

SG24-4640-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins.

Please either print out this questionnaire and complete it or complete it online. Then return it using one of the following methods:

- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

| Overall Satisfaction | ____ |
|----------------------|______|
| Organization of the book | ____ |
| Accuracy of the information | ____ |
| Relevance of the information | ____ |
| Level of technical detail | ____ |
| Value of illustrations | ____ |
| Grammar/punctuation/spelling | ____ |
| Ease of reading and understanding | ____ |
| Ease of finding information | ____ |
| Completeness of the information | ____ |
| INF quality | ____ |

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries: Yes____ No____
   Do you provide billable services for 20% or more of your time?

Are you in a Services Organization? Yes____ No____
Introduction to the current edition

Recent releases of OS/2 have introduced a number of so-called RAS enhancements and fixes to the System Debugging facilities. These enhancements have been released mainly in two major fix packs:

- Fix Pack 29 for Warp 3.0 and base Warp 4.0
- Fix Pack 35 for Warp 3.0 and Fix Pack 10 for Warp 4.0

All RAS enhancements are available with the base release of OS/2 Warp E-Server
This edition of the The OS/2 Debugging Handbook has been updated to include information on some of these new debugging facilities. For interim update 0.5 I have concentrated on updating the system structures and trace information for OS/2 Warp V4.0. Notes have been added throughout where the internal implementation differs from earlier releases.

Interim update 0.7 contains major updates to TRCUST, RASKDATA, SYSDUMP, TRAPDUMP, TRAPLOG and the System Trace Reference that reflect the latest RAS enhancements. It also includes additional information on the SYS317x exception popup message.

Other facilities that were introduced with these releases of OS/2, but are not yet covered, include:

- DTRACE - the low-level dynamic trace utility
- System trace enhancements
- Software trace
- PROC DumP, PDUMPSYS and PDUMPUSR utilities.
- PMDF enhancements
- TRACEFMT enhancements
- FFST/2
- Error Logging V2
- New REXX EXECs
- System Anchor Block

Future updates to the The OS/2 Debugging Handbook will include information on these topics together with updates on Warp SMP.

For further information on these new facilities the reader is referred to the README.DBG file, which is distributed with all fix packs that include the debugging enhancements.

--------------------------------------------

Approach to Problem Solving

In order to succeed at low-level program problem diagnosis, one must have several skills. None of these is particularly difficult, but many are foreign to today’s programmers.

At first, it will appear that each problem is solved with a different technique. Study of the methods used to solve problems yields the fact that the several skills are used as appropriate, virtually as subroutines, and without thought, by experienced analysts.

The intent of this material is to provide the basic knowledge and to illustrate each of the skills separately, to aid understanding. Trying to solve problems without the basic skills can be extremely frustrating, at best.

The fundamentals include knowledge of hardware operation, software conventions, and basic use of tools to display the data sought. Once the fundamentals are understood, it is time to begin using them to solve problems, because one can then concentrate on building the problem solving skill.

Application traps are perhaps the easiest problems to approach, so they are explained after the basic skills. Similarly, traps in privileged code are only incrementally more difficult.

Once some experience in solving traps has been gained, it is reasonable to extend one's skills by exploring reasons for waits and loops, collectively known as hangs, or to learn the additional functions provided by Symmetric Multi-Processor (SMP) systems, as well as the challenges in properly serializing them when needed.

--------------------------------------------

List of Necessary Skills

The fundamental skills are:

A good knowledge of how the hardware protection mechanisms work.
A good knowledge of what any instruction actually does.

A good knowledge of a few primary software conventions:

- How a stack is used and what information is in it.
- How to use the stack data for debugging.

How to use optional program documentation to get from a failing instruction to the actual line of the program which contains it.

How to find the program's variables in storage.

How to obtain the above documentation for some IBM languages.

How to collect a dump of a system at the point of failure.

How to use the available analysis tools.

How to determine the owner of a part of storage, and which processes have access to that storage.

And that's what this material is designed to teach!

--------------------------------------------

Collecting Documentation

If the problem can be reliably reproduced in a development environment, do it. This is the fastest way to get the problem fixed. When you cannot, try to get a good set of starting documentation.

It is possible to acquire and install a replacement for the OS/2 kernel which is the same as the one being replaced, except that it has debugging facilities and a debug interface to a serial port, COM2. If you install the wrong debug kernel, no one can predict the results. If you install the correct version, you will need to have a terminal emulation program (or ASCII terminal) to access the debug interface. The capabilities of this debug tool are essentially unlimited, and there is no protection from accidental entry errors. Its use is not a trivial task, nor one to be lightly undertaken.

It is often possible to collect enough information about a problem to diagnose its cause by creating customized trace entries specifically for that particular problem. For this to work well, the problem must be reproducible, and the trace buffer must be captured while the data gathered is still present.

Most people who have worked in a technical support role will agree that often the largest obstacle to solving a problem is collecting enough useful information about it. We will briefly discuss how to get enough useful data that problem solving can start in most cases. Be aware that frequently there will be some additional useful information, which can be gathered when the need for it is discovered, and that what is outlined here is not a complete list, by any means.

It is important to collect as complete a set of volatile data as possible from a single failure. If it is not gathered, it will be lost, perhaps requiring another occurrence of the problem in order to get needed information.

It is generally possible to use either an interactive debugger or a dump to diagnose either traps or hangs in an application.

For application problems, particularly traps, a good set of documentation includes the following:

- A statement of what sequence of events leads to the problem
- The trap screen, if a trap is involved
- A storage dump, with system trace data
- All the executable modules involved in the failure
- Optional application documentation, including:
  - all source files
  - .map files, produced by the linker
  - .lst and .cod or .asm files, produced by the compiler
The storage dump is the only thing which is volatile. The rest can be collected whenever the need is discovered. To collect the first item, perform the following steps:

**Note:** THIS WILL DRASTICALLY CHANGE OS/2 BEHAVIOUR WHEN A TRAP OCCURS. OS/2 WILL not CONTROL THE FAILURE, BUT WILL INSTANTLY AND IRREVOKABLY STOP THE SYSTEM, AND INITIATE A STORAGE DUMP. THERE WILL BE NO SHUTDOWN OF THE WORKPLACE SHELL, DATABASES, FILE SYSTEMS, (or lazy-write buffers,) OR ANYTHING ELSE. IT CAN BE AS DISRUPTIVE AS A POWER FAILURE. IT IS POSSIBLE TO LOSE FILES, OR PARTS OF FILES, but unlikely.

Prior to WARP: execute the command CREATEDD A: This will prepare a diskette for taking a dump. The diskette will work only once. This is not required for WARP, nor for later levels of 2.11. A quick way to discover if it is required is to read the prompt which asks for the diskette at the beginning of the process. If CREATEDD is required, the prompt asks for the diskette prepared by CREATEDD, otherwise it asks for a formatted diskette.

**Preparation:**

1. Save the current CONFIG.SYS
2. Edit CONFIG.SYS
   a. If the line is not already present, add a line which reads TRAPDUMP=ON
   b. add a line which reads TRACEBUF=63 to enable the system trace
   c. add a line which reads TRACE=ON to turn on the system trace
   d. optionally, add a line which reads TRACE=OFF,4,6,7
   e. optionally, turn LAZYWRITE off, so data goes directly to disk.
3. Locate some formatted diskettes to use for a storage dump.

Estimate about 2 Megabytes of RAM per diskette; usually one diskette more than that number is needed. For very large systems, estimate 1.5 meg per diskette. The dump process WILL NOT format.

4. Reboot the system so that the changes take effect.
5. Restore the original CONFIG.SYS, so you do not have to reboot an extra time to put things back to normal, after collecting the dump.

**Acquiring the storage dump:**

1. Cause the application to trap, that is, reproduce the problem.
2. Insert the CREATEDD diskette, if created, otherwise insert the first formatted diskette.
3. If you can read the screen, follow directions every time you hear one or more beeps.
4. If you cannot read the screen, you can still successfully get a dump, by listening for a beep. Insert the next diskette every time you hear a single short beep. When the dump is almost complete, there will be a very distinctively different series of beeps. At this point, reinsert the first diskette.
5. Very soon after the first diskette is reinserted, the dump will complete. Remove the diskette.
6. OS/2 will reboot automatically in most cases. Expect autocheck to run on HPFS drives during the boot.
7. Run CHKDSK on the drives as soon as convenient.

--------------------------------------------

**Hardware Architecture**

This section explains how the hardware operates in protected mode, what forms of protection exist, how they operate, and what happens when a program attempts to violate one or more of the protection mechanisms.

The three protection mechanisms in 32-bit OS/2 are:

1. Privilege
2. Description
3. Address mapping

**Note:** All three are active at all times when 32-bit OS/2 is running protected mode programs. Only address mapping is active when 32-bit OS/2 is running a VDM in V86 mode.

--------------------------------------------

Address Components

All addresses in x86 processors are composed of two parts:

**Note:** Addresses are usually written with a colon separating the two parts, for example, selector:offset.

1. A segment or selector
2. An offset

The offset part will be covered during the review of typical machine instructions, because it is straightforward, and the same in real and protected modes.

**Note:** These two parts are implicitly or explicitly specified by every instruction that references memory for either or both operands. Generally, the selector is implied and the offset is specified but there are exceptions to this.

--------------------------------------------

NEAR & FAR Addresses

Because there are two parts of an address and an item may or may not be in a current segment, there are two ways to specify the address of a data item.

A NEAR ADDRESS is an offset without specifying a selector. This is a very efficient way to address data because the overhead of loading a selector register and fetching the descriptor is avoided. The selector to use is implied, and is normally already loaded.

A FAR ADDRESS contains both a selector and an offset in protect mode. This is slower and more cumbersome because both address components must be specified as well as causing the overhead of altering a selector register. When a far address is displayed from storage (as two words), the offset will be seen in the left word, and the segment or selector in the right word.

A FAR ADDRESS contains a segment and an offset in real or V86 mode. The overhead is not so bad as in protect mode, because there are no descriptors to fetch when a segment register is loaded.

--------------------------------------------

Real Mode and V86 Mode

```
Real and V86 Modes
CS = Code Segment       SS = Stack Segment
DS = Data Segment       ES = Extra Segment

for 386 and later,
FS = another data segment  GS = another data segment
```

In REAL or V86 modes, say 'segment registers'
In PROTECT MODE, say 'selector registers'
Note: In real mode each segment register has a 16-bit number. The segment number is shifted left 4 bits, then added to the offset value. There is no checking of any kind.

\[
\begin{align*}
\text{DS}=1234, & \quad \text{offset}=5678 \\
12340 & \\
5678 & \\
\hline
179B8 & \\
\end{align*}
\]

Note: This is equivalent to any of the following:

- segment 179B, offset 8;
- segment 1790, offset B8;
- segment 1267, offset 5348;
- or many other possibilities.

Protected Mode

In protected mode, all storage is described by the hardware, using tables maintained by the software. The description includes the location, and size of the storage segment, as well as the type of storage. The storage type further constrains how it may be used.

This section concentrates on the selector part of the address because the offset is handled in a very simple and consistent fashion once the memory segment has been located and the validity of the access has been verified.

Descriptors

A selector specifies a descriptor, which describes a memory segment. The attributes described include the base or starting address of the memory segment, the size of the segment and what accesses are allowed.

Protected mode addressing in a 386 or later begins with Descriptor Tables which are described by hardware registers. There are three Descriptor Tables, each of which is discussed below after supplying the format of individual descriptors. The tables contain the descriptors and the descriptors are selected by an interrupt number or by the content of a selector register.

An application descriptor is required for all accesses to instructions and to data. For most segments, the limit is the largest valid offset. If the offset is larger than the limit, a general protection exception occurs. The exception to this rule occurs for data segments which are 'expand down'. In this case, the offset must be greater than the content of the limit field. The system stack (ring 0) is an example of an Expand Down segment.

To find the linear address of the data element, the processor adds the offset (obtained from the instruction) to the base address of the segment. That's the end of the discussion for offsets!

There are three distinct kinds of data recognized by the processor:

- Stack, which holds temporary data, parameters and return addresses
- Code, which is instructions for the processor to execute
- Data, which is used to hold data which is available for longer than the lifetime of any one function or routine.

The primary distinction between stack and data is that data segments begin at offset zero and expand upward (to the limit) while stack segments begin at the highest offset and expand downward (to just greater than the limit). Many language implementations use data segments for their stack, which is perfectly acceptable, but it makes it impossible to ‘grow’ the stack.

The descriptor for a memory reference is found by using the appropriate selector as the index to a table or, if you ignore the 3 lower bits, as an offset to the table, since descriptors are 8 bytes long.
Selector Format

In protect mode, a Selector has 3 fields:

1. **Index**, the left 13 bits, bits numbered 15-3
   - This is an index into a descriptor table

2. **Table indicator**, one bit, bit number 2
   - 0 means GDT
   - 1 means LDT

3. **RPL**, the right 2 bits, numbered 1 & 0.
   - Requested Privilege Level.
   - Perceived as a two bit value, range 0 to 3: 00=most privileged, or ring 0; 11=least privileged, or ring 3.

**Note:** The position of the bits makes a selector (with its 3 low order bits turned off) the offset into the table.

```
+--------------------------------------+--+-----+
|         I N D E X                     TI  RPL | fields
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0  bit numbers
```

Privilege Levels

The point of privilege levels is to prevent a program from accessing a storage object that is more privileged than the program itself. Generally, this means that application programs are not able to access storage used by supervisory programs in any way. This also means it is safe to keep descriptions of storage used by the system in a descriptor table that can be accessed by applications, because the application cannot use those descriptors.

There are actually 3 distinct privilege levels associated with every storage access, and testing privilege level is a two-step process. The privilege level used to access a storage operand is the less privileged of CPL and RPL. The first step is to determine the actual privilege level with which to attempt the access. The second step is to compare the privilege level of the storage object (from the descriptor) to the result of the first step.

- **DPL** Descriptor Privilege Level.
  - Bits 45 & 46 of descriptor.

- **RPL** Requested Privilege Level.
  - 2 low order bits of selector.

- **CPL** Current Privilege Level.
  - 2 low order bits of CS.

A more privileged (lower numbered) program may access the storage objects of a less privileged program. This is how the operating system returns structures and fills in data areas for an application.

Any attempt by a less privileged (higher numbered) program to access in any way a storage object which is more privileged generates a general protection exception.
Descriptor Tables

There are three tables which hold descriptors.

The three tables are:

1. The Global Descriptor Table or GDT, describes memory objects which are accessible to all processes.
   The GDT is located by means of a hardware register called the GDTR which contains the linear address and length of the GDT.

2. The Local Descriptor Table or LDT, describes memory objects which are unique to one process or are shared among a few processes by design.
   The LDT is located by means of a hardware register called the LDTR which contains a selector. The descriptor referenced by this selector must be a system descriptor which describes an LDT.

3. The Interrupt Descriptor Table or IDT, has gates that specify interrupt handler entry points.
   The IDT is located by means of a hardware register called the IDTR which contains the linear address and length of the IDT. The interrupt number is used to index into this table when an interrupt occurs.

Descriptor Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Tells what kind of object is described</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application types:</td>
<td>Code, Data</td>
</tr>
<tr>
<td>System types:</td>
<td>LDT, TSS, Call Gate, Irpt Gate</td>
</tr>
</tbody>
</table>
### Base
Linear address of object

### Limit
Defines the size of a storage object

### DPL
Privilege level defines which ring(s) can access the described object

---

**LIMIT 00-15**  **BASE 0-23**  **TYPE S DPL P LIMIT 16-19**  **FLAGS BASE 24-31**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>
| byte offsets

Display a descriptor with 'DB' to see it in this form.

#### Notes:
- **TYPE** is what kind of object is described
- **S** is descriptor category; 0=system, 1=code or data
- **PL** is privilege level of object described
- **P** is the present bit; 1=present, 0=not present

---

### Descriptor Flags

**Bit 55**  Granularity: (G) 0=limit is in bytes, 1=limit is in 4K pages

**Bit 54**  Default address and operand (D/B) size: 0=16 bit, 1=32 bit

**Note:**
- In code segment this bit (called the D bit) governs the default address and operand size.
- In a data segment this bit is ignored.
- In a stack segment this bit (called the B bit) determines whether SP (B=0) or ESP (B=1) is used by instructions that implicitly reference the stack, for example PUSH, POP, CALL and RET.

**Bit 53**  Reserved, must be zero.

**Bit 52**  Unused by hardware, used by OS/2 to indicate **UVirt**

**Bit 47**  Present: (P) 1=segment is present, 0=segment is not present

**Bits 46 & 45**  Privilege Level: 00=most, 11=least

**Bit 44**  Segment type: 0=system segment, 1=application segment

**Bit 40**  Accessed: (A) 0=not accessed, 1=accessed

**Note:** If application segment, ( Bit 44 = 1 ), used to store program code and data.
  - Bit 43=0 is Data Segment
  - Bit 42: Expansion: 0=Expand Up, 1=Expand Down
  - Bit 41: Writeable: 0=Read Only, 1=Read/Write
  - Bit 43=1 is Code Segment
  - Bit 42: Conforming: 0=Non-conforming, 1=Conforming
  - Bit 41: Readable: 0=Execute Only, 1=Read/Execute

**Note:** If system segment, ( Bit 44 = 0 )

**Bits 39-42**  Type of segment
00      RESERVED
01      Available 286 TSS ( 16-bit )
02      LDT
03      Busy 286 TSS ( 16-bit )
04      286 Call Gate ( 16-bit ) ( Parm Count is words )
05      Task Gate
06      286 Interrupt Gate ( 16-bit )
07      286 Trap Gate ( 16-bit )
08      RESERVED
09      Available 386 TSS ( 32-bit )
10      RESERVED
11      Busy 386 TSS ( 32-bit )
12      386 Call Gate ( 32-bit ) ( Parm count is doublewords )
13      RESERVED
14      386 Interrupt Gate (32-bit )
15      386 Task Gate ( 32-bit )

--------------------------------------------

Descriptor Table Summary

There are three descriptor tables at any instant.

1.  Global Descriptor Table
    located via GDTR
    1 per system
    accessible to all processes
    describes objects common to all processes

2.  Local Descriptor Table
    LDTR is selector
    GDT Descriptor Locates LDT
    1 per process except for VDMs in which multiple LDTs are possible.
    describes data unique to one process

3.  Interrupt Descriptor Table
    located via IDTR
    1 per system except for VDMs in which multiple IDTs are possible.
    describes interrupt routine entry points

--------------------------------------------
The Selector Registers

Each selector register appears to be 16 bits long. The six application selector registers and a brief description of the use for each follow:

SS: Stack Selector, specifies the descriptor used for stack references.
CS: Code Selector, specifies the descriptor used for instruction references.
DS: Data Selector, specifies the descriptor used for most data references.
ES: Extra Selector, specifies another descriptor used for data references.
FS: This is a selector which can be used for data references if explicitly specified.
GS: This is a selector which can be used for data references if explicitly specified.

The two system selector registers and a brief description of the use for each follow:

LDTR: The LDT register selects the LDT descriptor from the GDT.
TR: The Task Register selects the descriptor used for the TSS.

When Checking Is Done

When a program moves data into a selector register, that data becomes a selector and the processor fetches the content of the appropriate entry from the specified table into onboard registers which are not accessible to the programmer. The processor verifies the validity of the attempted access to the memory whenever a selector register is updated. This makes the protection overhead occur as part of the instruction which modifies a selector register, but eliminates it for further use of the selector.

Note: If the RPL of the SS register is not the same as CPL, or if an attempt is made to move the null selector into SS, a general protection exception occurs.

Note: The first descriptor in the GDT is reserved, by definition, and cannot be used. Any selector which would reference it is called the NULL SELECTOR; possible values are 0000, 0001, 0002, and 0003.

By definition, the null selector may be placed in DS, ES, FS, or GS, but any attempt to form an address with it is a general protection fault.

Note: The LDTR is a register that contains a selector. It can be accessed only by privilege level 0 instructions. It must contain a selector that references the GDT, and a descriptor whose "type" is LDT.

Note: It is not unusual for a GDT selector to describe the same storage as an LDT selector does. In OS2 2.x, application selectors in the GDT happen to describe one 448 Meg segment, not just a 64K segment like the LDT selectors describe. The linear address assigned to each LDT descriptor is extremely convenient for changing one form of an address to another, called thunking, which will be discussed later.
Descriptor Examples

These examples come from DUMP1, which is used for several exercises.

```
DL 7 37
0007 Data  Bas=ac6d7000 Lim=000000ff DPL=3 P  RO
000F Code  Bas=00010000 Lim=00002e77 DPL=3 P  RE A
0017 Data  Bas=00020000 Lim=0000290f DPL=3 P  RW A
001F Data  Bas=00030000 Lim=000018af DPL=3 P  RW A
0027 Data  Bas=00040000 Lim=0000030a DPL=3 P  RW A
002F Data  Bas=00050000 Lim=000000ff DPL=3 P  RW
0036 Data  Bas=00060000 Lim=000000ff DPL=2 P  RW A
DL BECF
bece  Code  Bas=17d90000 Lim=00000010 DPL=2 P  RE A
DL BFD7 BFEF
bfdf  Data  Bas=17fa0000 Lim=0000ffff DPL=3 P  RW A
bfdf  Data  Bas=17fb0000 Lim=0000ffff DPL=3 P  RW A
bfee  Code  Bas=17fd0000 Lim=00000aa2 DPL=2 P  RE A
DG 20 78
0020 Data  Bas=ffe5b000 Lim=000003ff DPL=0 P  RW UV
0028 LDT  Bas=ac6d7000 Lim=000000ff DPL=0 P
0030 Data  Bas=ffe09de4 Lim=0000421b DPL=0 P  RW ED A UV
003b Data  Bas=ffe4be2c Lim=00000073 DPL=3 P  RW
0040 Data  Bas=ffe5be00 Lim=000003bf DPL=0 P  RW UV
004a Data  Bas=00000000 Lim=1bffffff DPL=2 P  RW A G4k BIG UV
0053 Data  Bas=00000000 Lim=1bffffff DPL=3 P  RW A G4k BIG UV
005A Data  Bas=00000000 Lim=1bffffff DPL=2 P  RE C A G4k C32 UV
0063 Data  Bas=00000000 Lim=1fffffff DPL=3 P  RW G4k BIG UV
006B Data  Bas=00000000 Lim=1fffffff DPL=3 P  RW A G4k BIG UV
0070 Data  Bas=ffe22000 Lim=000074e4 DPL=0 P  RO A
0078 Data  Bas=ffe22000 Lim=000074e4 DPL=0 P  RW
DG 148 L 4
0148 Code  Bas=ffe39000 Lim=00009262 DPL=0 P  RE A
0150 Code  Bas=ffe39000 Lim=0000e137 DPL=0 P  RE A
0158 Data  Bas=00000000 Lim=fffffffe DPL=0 P  RW A G4k BIG
0160 Code  Bas=00000000 Lim=fffffff DPL=0 P  RE A G4k C32
```

The top section of the above output was created by entering the command **DL 7 37**

By inspecting the type, base, and limit fields in the above output, we can see the following about the descriptor referenced by 002F:

The storage is described as data having a base, or linear, address of 00050000. The linear address is not normally written with leading zeros. If there were any chance that the address might be mistaken for physical, a percent sign would be used, for example, `%50000`. The limit is FFF, which means that the segment is 4K, or 1000(hex) long. The privilege level is 3, the segment is present, and the flags indicate Read/Write storage. It has NOT been accessed, because the ‘A’ flag is not present, and OS/2 no longer uses this flag; once set by the hardware, it remains set.

Examples related to privilege level protection follow below:

<table>
<thead>
<tr>
<th>CS:IP</th>
<th>CPL</th>
<th>DS:xxxx</th>
<th>RPL</th>
<th>lesser privilege CPL &amp; RPL</th>
<th>DPL (from descriptor)</th>
<th>Access allowed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>17:xxxx</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>16:xxxx</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>14:xxxx</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>37:xxxx</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>36:xxxx</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>34:xxxx</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>43:xxxx</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>42:xxxx</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>000F:xxxx</td>
<td>3</td>
<td>40:xxxx</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>BECE:xxxx</td>
<td>2</td>
<td>17:xxxx</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>BECE:xxxx</td>
<td>2</td>
<td>16:xxxx</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
</tbody>
</table>
In each case, as you read across you will see that CPL comes from the value of the CS register, RPL comes from the two low-order bits of
the selector, and DPL comes from the descriptor. The column titled 'lesser privilege' is calculated remembering that higher numbers are
lower privilege. The final column is obtained by following the access rules, a short way back.

Exercise 1: Selectors and Descriptors

Objectives:

1. Learn how to load a dump for analysis
2. Introduction to the dump formatter
3. Learn how to display descriptors

Start the lab at a full-screen or windowed command prompt.

A full-screen session is faster, but a windowed session can be made 100 lines high by entering

MODE CO80,100 This can be very useful, because you can look back quite a ways by using the scroll bar.

Change to directory CLASSES\UTIL

Make diskette one by typing OS2IMAGE ..\IMAGES.162\LAB01.001 A:

Make diskette two by typing OS2IMAGE ..\IMAGES.162\LAB01.002 A:

Load the dump into a new file which will be named DUMP1.DMP by typing

DCOMP A: X:\DUMP01.DMP and pressing enter, then following the prompts.

When the dump is loaded, it should have a file size of 4194816.

Start the dump formatter by typing DF_RET X:\DUMP01.DMP,

or by DF_RET ..\DUMPS.162\DUMP01.DMP

You should see 6 or 7 informational lines at the top, followed by a pair of lines which start "Slot", and "**0023#", followed by a set of registers. *** We are not yet concerned with any of these. ***

You should get a prompt, which is the character "#".

**Note:** You can always document what you are thinking by simply typing it in as an evaluation for the dump formatter to perform. You can access the evaluation function by typing ? followed by whatever you want echoed to the screen and to the log. You can also type in ? and any expression to have it evaluated and output in hex, decimal, octal(!), binary, character, and boolean forms.

**Note:** ? by itself is a simple request for what commands are recognized.

Use the dump formatter to look at descriptors and answer these questions.

The dump formatter is NOT case sensitive.
Descriptors may be displayed using "DG" or "DL", followed by the selector. Try it both ways for several selectors, such as F, 160, DFF, 158.

Use the miniature command reference in the back of the student guide, if necessary.

There are a great many things we will NOT do in this exercise. We are using only a tiny part of the dump formatter’s capabilities for this class. For example, we will ignore the IDT in this class; one can enter “DI” followed by the interrupt number to see the descriptor from the IDT.

Questions to answer:

1. Which table contains the descriptor data for selector 000F?
2. Which command is preferred to display only the descriptor for 000F?
3. What alternative command will also display only the descriptor for 000F?
4. What type of memory is described by selector 000F?

Hint: It is one of the first things displayed in the output for each descriptor.

5. What is the largest valid offset within segment 000F?
6. What is the size of segment 000F?

Hint: Not quite the same as the previous answer.

7. What is the linear (virtual) address of segment 000F?
8. What privilege level is segment 000F?
9. What is the Requested Privilege Level of selector 000F?

Hint: RPL is not in the descriptor.

10. What is the type and limit of segment 0017?
11. What is the linear (virtual) address of segment 0017?
12. Which table contains descriptor 0017?
13. Will the application program be able to access the segment selected by 000F?

Explain __________________________________________

14. Will the program be able to store into segment 000F?

Explain __________________________________________

15. Will the application program be able to access storage using selector 0037?

Explain __________________________________________

16. Will the program be able to write into storage using selector 38?

Explain __________________________________________

17. Will the program be able to write into storage using selector 0007?

Explain __________________________________________

18. Enter the following command: DG 70 L 2

Compare and contrast the base, limit, privilege level and flags for each.

19. Enter the following command: DG 5A;DG 5B

Compare and contrast the base, limit, privilege level and flags for each.

20. Enter the following command: DG 28;DL 7

Compare and contrast the base, limit, privilege level and flags for each.
Address Mapping

This section describes the method used to transform addresses from linear addresses to physical addresses.

Paging Overview

OS/2 V2 uses paging in addition to the above logical addressing. Paging is a mechanism which converts linear addresses to physical addresses and allows a consistent size (4k) to be moved back and forth to auxiliary storage (SWAPPER.DAT) when the demand for virtual memory exceeds the physical memory installed on the machine. Another hardware register, Control Register 3 or CR3, is used to locate a page directory which contains table entries that locate page tables. The page tables are used to locate the physical memory where the data really resides. Physical memory is sometimes referred to by page number. A page number is simply the twenty high-order bits of an address. The twelve low-order bits of a page address are all zero. One can convert a page number to an address by simply appending three hex zeros to it.

The result of combining a segment number and an offset, or the addition of an offset to the base address from a descriptor, is a linear address. Under OS/2 1.x, these would be physical addresses. Under OS/2 2.0 and following, these are linear, or virtual addresses.

The picture below shows how linear addresses are converted to physical addresses. Only the top line in the picture below is a linear address - the rest are physical.

The ten high order bits of the linear address are used to index into the Page Directory which has the twenty high order bits of the page table's physical address (page number). The next ten bits of the linear address are used to index into the page table. The twenty high order bits of the page frame's physical address (page number) are retrieved. The twelve low order bits of the linear address are also the twelve low order bits of the physical address. Therefore, the physical address is the twenty bits from the page table entry, followed by the 12 low-order bits from the linear address.

Page Table Entries

The page directory entries are identical to the page table entries.
Each entry is 4 bytes, making 1K entries in each page table.

**Bits 31-12** Physical Address of page, or Page Frame Address

**Bits 11-09** Ignored by hardware, used by OS2. See Note.

**Bits 08-07** Reserved, must be zero

**Bit 6** Dirty (D) 0=not changed (clean), 1=changed (dirty)

**Bit 5** Accessed (A) 0=not accessed, 1=accessed

**Bit 4** Page Cache Disable 0=allow cache use, 1=bypass cache

**Bit 3** Page Write-Through 0=cache write-into, 1=write through to RAM

**Bit 2** Supervisor (S/U) 0=Supervisor (PL=0,1,2), 1=user (PL=3)

**Bit 1** Write enable (RO/RW) 0=Read Only, 1=Read/Write

**Bit 0** Present (P) 0=not present, 1=present

**Note:** The left 5 hex digits of the entry are the left 5 hex digits of the physical page; while the right 3 hex digits are mostly flags.

**Note:** If Bit 0 is zero, ( page invalid ) the remaining bits are NOT inspected by the hardware. OS/2 uses them to identify the virtual page associated with this address.

**Note:** Bits 09 and 10 are used to track the state of the page frame. Three of the possible four combinations are used:

- 0 - Pageable
- 1 - UVirt
- 2 - Resident

Page Table Contents

To look at the contents of the page directory and page table(s), use the command **DP**, followed by the address of interest.

```
DP F:0
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00000000* 001e0 frame=0009e 0 0 c A U r P pageable
%00000000 0009e frame=0009e 0 0 c A U r P pageable
%000011000 vp_id=012ae 0 0 c u U r n pageable
%000012000 00292 frame=00292 0 0 c A U r P pageable

DP 17:0
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%000020000* 001e0 frame=00181 0 0 D A U W P pageable
%000020000 00181 frame=00181 0 0 D A U W P pageable
%000021000 003d4 frame=003d4 0 0 D A U W P pageable
%000022000 0005a frame=0005a 0 0 D A U W P pageable

DP 1F:0
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%000030000* 001e0 frame=003ae 0 0 D A U W P pageable
%000030000 003ae frame=003ae 0 0 D A U W P pageable
%000031000 001b5 frame=001b5 0 0 D A U W P pageable

DP 27:0
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%000040000* 001e0 frame=00052 0 0 c A U W P pageable
%000040000 00052 frame=00052 0 0 c A U W P pageable

DP 2F:0
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%000050000* 001e0 frame=00075 0 0 D A U W P pageable
%000050000 00075 frame=00075 0 0 D A U W P pageable
```
In each case, the first line of output is the data from the page directory.

The field labelled 'frame' is the physical page frame which holds the data at the referenced address.

The 'vp id' is the virtual page identifier for the entry %11000.

'Dc' is Dirty or Clean. 'Au' is Accessed or unaccessed.

'Us' is User (Ring 3) or supervisor (Rings 0 & 2).

'rW' indicates read-only or Writeable. 'Pn' indicates Present or not-present.

---

### Data Format in Storage

Data format is least significant byte at lowest address!

This arrangement is not intuitive for many people, because when you read bytes, the data placement seems reversed. The tools will let you display storage as bytes, words, and doublewords; the data will be re-arranged to suit the format requested. This can be good or bad.

For example:

- **DB 1F:1608 L 20**
  - 001f:00001608 BOOKSHELF=C:\OS2
  - 001f:00001618 COMSPEC=C:\OS2\BOOK;

- **DA 1F:1608**
  - 001f:00001608 BOOKSHELF=C:\OS2\BOOK;

- **DD 17:0 L 10**
  - 0017:00000000 002 00 0003 00005 0007 0000b 0000d 0011 0013
  - 0017:00000001 0017 001d 001f 0025 0029 002b 002f 0035 0035
  - 0017:00000002 003b 003d 0043 0047 0049 004f 0053 0059 0059
  - 0017:00000003 0061 0065 0067 006b 006d 0071 007f 0081 0081

- **DD 17:1 L 10**
  - 0017:00000000 004c 0050 0700 0b00 0d00 1100 1300 1700
  - 0017:00000001 0100 1f00 2500 2900 2b00 2f00 3500 3b00
  - 0017:00000002 03d003b 0470043 04f0049 0530053 0590053
  - 0017:00000003 0650061 06b0067 071006d 083007f

- **DD 17:2 L 10**
  - 0017:00000000 05000000 0b000700 11000d00 17001300
  - 0017:00000001 1f001d00 29002500 2f002b00 3b003500
  - 0017:00000002 43003400 49004700 53004f00 61005900
  - 0017:00000003 67006500 6d006b00 7f007100 89008300

- **DD 17:3 L 10**
  - 0017:00000000 00500003 00b00070 0011000d 0170013
  - 0017:00000001 01f001d 00290025 002f002b 003b0035
  - 0017:00000002 043003d 0490043 053004f 0610059
  - 0017:00000003 0670065 06d006b 07f0071 0890083
Exercise 2: Paging, Addresses, Data

Objectives:
1. Reinforce the knowledge from exercise 1
2. Learn how to display page table data
3. Learn how to convert a logical address to a linear address
4. Learn how to convert a linear address to a physical address
5. Learn how to display storage as ASCII, bytes, words, and doublewords.

Startup directions:
1. Start the dump formatter by typing `DF_RET ..\DUMPS.162\DUMP01.DMP`
2. You should see the standard startup messages.
3. The initial register display is what the application registers were at the time the application (ring 3) program trapped.
4. You can see these at any time by entering the ".R" command.
5. Use the dump formatter to look at the dump and answer these questions.

Note: Paging data may be displayed using the "DP" command, followed by the address.

Note: The dump process DESTROYS the first entry of the page directory. You will get quite confused if you try to follow the hardware method to look at paging information for addresses 0 - 3FFFFF.
If you must, use the ".N" command to find "savepage", which will tell you the physical address of the page table for that address range.

This may well be the last time you use a physical address in an OS/2 debugging session. With the notable exceptions of physical memory management and physical device drivers, OS/2 is almost completely unaware of physical addresses. The 32-bit virtual address, also called a linear address, and a 'flat' address, is what is used in general throughout OS/2.

Assuming these registers, answer the following questions:

```
eax=0000c8cf ebx=00002910 ecx=000000df edx=00000000 esi=00000030 edi=00000060
eip=000000be esp=000014be ebp=000014e6 iopl=2 rf -- -- nv up ei pl zr na pe nc
cs=000f ss=001f ds=001f es=0017 fs=150b gs=0000 cr2=00000000 cr3=001a7000
```

1. What are the base and limit fields for selector 000F? (the base is the linear address...)
2. How many 4k pages are in this segment? Hint: Look closely at the limit field.
3. How many physical pages are allocated for the virtual memory segment starting at F:0?

Hint: DP 0F:0 or DP %10000

4. Why are the above two answers different?
5. What is the physical address of the data at F:0?

Observation: You now have three ways to address the data.

a. Real or V86 (&selector:offset)
b. Logical (#selector:offset)
c. Linear (%address)
Physical (%%address)

We will now display the same storage many ways, to confirm we know how.

6. What is the command to display the storage at SS:BP in words using a logical address?
7. What is the command to display the storage at SS:BP in words using a linear address?
8. What is the command to display the storage at SS:BP in words using a physical address?

For each of the following, study the results until you understand.

9. Display the data at 7:0 as bytes, and words.
10. Display the data at 7:1 as bytes and words.
11. Display the data at 7:0 and 7:1 as words.
12. Display the data at 7:0 as words and doublewords.
13. Display the data at 1F:15C6 as bytes and ASCII. Also look at 1F:15DA as bytes and ASCII.

----------------------------------------------------------

Instruction Set

This section discusses the '86 registers & some common instructions from the instruction set.

----------------------------------------------------------

Register Review

Registers discussed so far:

CR3  32-bit physical address of the Page Directory
IDTR 32-bit linear address of IDT, 16-bit size of IDT
GDTR 32-bit linear address of GDT, 16-bit size of GDT
LDTR 16-bit selector for an entry (type 2) in the GDT
SS 16-bit selector, used for stack operations
CS 16-bit selector, used to locate instructions
DS 16-bit selector, used to locate data, generally the default
ES 16-bit selector, used to locate data, string destination
FS 16-bit selector, used to locate data explicitly
GS 16-bit selector, used to locate data explicitly

----------------------------------------------------------

Execution

386 execution consists of the classic pattern of fetching an instruction from memory and executing it, then repeating the process. The instructions are always found in a code segment accessed via the descriptor designated by the selector in the CS register. The current privilege level of the program is contained by the two low order bits in the CS register. The offset of the next instruction is contained in the
instruction pointer, (IP or EIP) which is incremented as each instruction is fetched. The 386 and following generations recognize a great number of instructions, but compilers generate a very small subset of the whole instruction set. Much of that subset will be discussed here. If you cannot ascertain what an instruction does when you encounter it, look it up in the appropriate reference manual. Instructions are generally executed sequentially, and the processor attempts to fetch instructions well in advance, to increase execution speed. The flow of control departs from sequential when a jump, call, return, interrupt or interrupt return is encountered. Jumps are conditional or unconditional. Conditional jumps are used to implement decisions and contain a relative offset which is combined with IP by signed addition to cause a different instruction in the same segment to be executed next. Calls, returns and unconditional jumps come in two varieties: NEAR and FAR. The NEAR variety update only IP and leave CS untouched. The FAR variety update both CS and IP and are potentially quite complex. CALL, RETurn and interrupts require a stack. Most instructions reference the registers.

General Registers

<table>
<thead>
<tr>
<th>EAX</th>
<th>ALL 32 BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(part of EAX)</td>
<td>AX</td>
</tr>
<tr>
<td>(part of AX)</td>
<td>AH</td>
</tr>
<tr>
<td>(part of AX)</td>
<td>AL</td>
</tr>
</tbody>
</table>

Registers EBX, ECX, and EDX also subset in the same way. There are two byte-sized pieces, which can be collectively referenced as a word-sized item.

<table>
<thead>
<tr>
<th>EIP</th>
<th>ALL 32 BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(part of EIP)</td>
<td>IP</td>
</tr>
</tbody>
</table>

IP and EIP are always offsets into CS. They always contain the address of the next instruction to execute.

<table>
<thead>
<tr>
<th>ESP</th>
<th>ALL 32 BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(part of ESP)</td>
<td>SP</td>
</tr>
</tbody>
</table>

SP and ESP are always offsets into SS. They contain the address of the last item pushed into the stack.

REGISTERS EBP, ESI, and EDI also subset in this way. They have no 8 bit parts.

Machine Instructions

There are several fields which may be present in an instruction. Additionally, there are a few easy-to-learn generalities which will make understanding what an instruction does much easier. Data definitions will not be covered here. There are many fields possibly present in an instruction.

1. The label.
The label is optional, but must be first. It is followed by a colon. It is used so the programmer can refer to the instruction symbolically. A label does not require an instruction.

Labels which are ‘Public’ become symbols at link time.

2. The mnemonic operation code, or opcode, is next. It defines what operation will be attempted, and therefore what operands need to be specified. Instructions have zero to three specified operands; many instructions also imply operands.

3. The operands are next, separated by commas. The first operand is always the result, or target, of the operation.

An operand may be a value, a register, or storage. When the operand is a value, it is called ‘immediate’, because the operand is immediately available if the instruction has been fetched. When a register is named, it is the operand. If an expression is contained in brackets, it is evaluated and the result is used as a offset into some segment.

A storage operand is in some segment by default. Data references default to the data segment, or DS, unless (E)BP or (E)SP are present in the address expression. In this case, the default segment is the stack segment (SS). (E)IP is ALWAYS in the CODE segment (instructions). (E)SP is ALWAYS in the STACK segment (data). (E)BP is USUALLY in the STACK segment (data).

The default segment can usually be overridden by specifying the selector as part of the address, for example, DS:[BP+8].

You will come across helper words within operands, such as "byte", "word", and "dword" which are there to remind you of the size of the data item referenced. You will also come across the helper word "ptr", which is to remind you that the addressed data is in storage, and that the offset, in brackets, is a pointer to the data.

4. The last item you may find is an optional comment. A comment is preceeded by a semicolon. Anything following is a comment. Comments are sparse in the output of the ‘Unassemble’ command.

The debug kernel will use a comment to identify a breakpoint.

Both the debug kernel and the dump formatter will supply a symbol anytime a number matches the symbol in an active file.

--------------------------------------------

Typical Instructions

MOV   CL,DH

The opcode is ‘MOV’, the first operand is the CL register, and the second operand is the DH register. This instruction will copy (MOVe) all 8 bits from the DH register to the CL register.

MOV   DX,8

The opcode is ‘MOV’, the first operand is the DX register, the second operand is the immediate value of 8. This instruction puts the value 8 into the DX register.

MOV   EBP,ESP

Again, the opcode is ‘MOV’, and the instruction will copy all 32 bits of ESP into EBP.

MOV   AX,BX

You should be able to tell by now that this instruction will copy 16 bits from BX to AX. Note that instructions which reference only registers are extremely unlikely to cause an exception.

MOV   AX,word ptr [BX]

This instruction is different from the one above because there are brackets around the second operand. This means that the operand, BX in this case, is in storage, and the BX register holds the offset into the DS segment. If BX is outside the limit of the DS segment, a general protection fault will occur.

MOV   word ptr [BX],AX
This instruction is similar to the preceding one, but moves data into storage, rather than from storage. The same exceptions might occur, and if the DS segment is read-only, this instruction would also fail.

MOV
word ptr ES:[BX],DI

This is an example of overriding the default segment, DS, by explicitly specifying that the offset in the BX register applies to the ES segment.

ADD
word ptr DS:[BP],AX

This would add the 16 bits from AX into storage at DS:BP, developing the sum directly in storage. The override is needed because the use of BP means that the default segment is SS.

DEC
word ptr [BP-2]

Some instructions have only one operand. In this case it is in storage at an offset calculated by subtracting 2 from the BP value, in the segment defined by the SS register, because BP is used.

Also SUB, CMP, AND, OR, XOR, XCHG, INC, SHL, etc.

It is extremely common for 16-bit code to use FAR addresses. When they are in storage, it would require several instructions to get a FAR address into the registers, if it were not for several instructions whose purpose is specifically to fetch a FAR address from storage into a selector and another register. These instructions may be recognized by the opcode, which is the letter 'L' followed by a selector register name other than CS. The apparent first operand is the general, base, or index register which will hold the offset part of the far address. Both registers will be loaded, with the first operand coming from the address specified, and the selector coming from the following word.

LES
BX,dword ptr [BP+6]

This instruction loads BOTH BX and ES. BX comes from BP+6 and ES comes from BP+8, both in the stack segment.

LDS
SI,dword ptr [BP-12]

This instruction loads BOTH SI and DS, SI is loaded from BP-12 and DS is loaded from BP-10.

LEA
EDI,[EBP+ECX*4-12]

Load Effective Address DOES NOT actually reference storage. Instead, once the offset has been calculated, it is put into the target register, EDI in this case. Address expressions like this are possible, but not often seen while actually debugging. The scale factor can be 1, 2, 4, or 8; not any arbitrary value

--------------------------------------------

The System Flags (EFLAGS Register)

The flags, which are contained in the EFLAGS register, not only control system operation, but also hold the result of instructions such as CMP (compare). At times, you will find the flags have been copied to a register, or to memory. The following figure gives the format of the flags in such cases:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Hex</th>
<th>Flag name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>00040000</td>
<td>AC</td>
<td>Alignment Check, if the alignment mask is 1 (CR0).</td>
</tr>
<tr>
<td>17</td>
<td>00020000</td>
<td>VM</td>
<td>V86 mode. Turned on for Virtual DOS Machines.</td>
</tr>
<tr>
<td>16</td>
<td>00010000</td>
<td>RF</td>
<td>Resume Flag. Suppress debug exceptions for 1 instruction.</td>
</tr>
<tr>
<td>14</td>
<td>00004000</td>
<td>NT</td>
<td>Nested Task. Involved with hardware task switching.</td>
</tr>
<tr>
<td>13/12</td>
<td>3000</td>
<td>IOPL</td>
<td>The least privileged code which has unrestricted I/O access.</td>
</tr>
<tr>
<td>11</td>
<td>0800</td>
<td>OF</td>
<td>Overflow. An arithmetic result does not fit.</td>
</tr>
<tr>
<td>10</td>
<td>0400</td>
<td>DF</td>
<td>Direction of string instructions. 0=up, 1=down.</td>
</tr>
<tr>
<td>09</td>
<td>0200</td>
<td>IF</td>
<td>Interrupt flag. 1=enabled, 0=disabled.</td>
</tr>
<tr>
<td>08</td>
<td>0100</td>
<td>TF</td>
<td>Trap flag. Generate a debug exception after each instruction.</td>
</tr>
<tr>
<td>07</td>
<td>0080</td>
<td>SF</td>
<td>Sign. 1=minus, 0=plus.</td>
</tr>
<tr>
<td>06</td>
<td>0040</td>
<td>ZF</td>
<td>Zero or Equal. 1=zero result, 0=non-zero result.</td>
</tr>
<tr>
<td>04</td>
<td>0004</td>
<td>AF</td>
<td>Auxiliary flag. Used in BCD arithmetic.</td>
</tr>
<tr>
<td>02</td>
<td>0004</td>
<td>PF</td>
<td>Parity flag. 0=even, 1=odd.</td>
</tr>
<tr>
<td>00</td>
<td>0001</td>
<td>CF</td>
<td>Carry flag. 0=no carry, 1=carry.</td>
</tr>
</tbody>
</table>

--------------------------------------------
Unassembled Instructions

U CS:IP-22 IP-18
000f:0000009c f1       db       f1
000f:0000009d 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-23 IP-18
000f:0000009b f7f1     div      cx
000f:0000009d 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-24 IP-18
000f:0000009a ee       out      dx,al
000f:0000009b f7f1     div      cx
000f:0000009d 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-25 IP-18
000f:00000099 56       push     si
000f:0000009a ee       out      dx,al
000f:0000009b f7f1     div      cx
000f:0000009d 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-26 IP-18
000f:0000009b 8b56ee   mov      dx,word ptr [bp-12]
000f:0000009c f7f1     div      cx
000f:0000009e 8b56ee   mov      dx,word ptr [bp-12]
000f:0000009f 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-27 IP-18
000f:00000097 ec       in       al,dx
000f:0000009b 8b56ee   mov      dx,word ptr [bp-12]
000f:0000009c f7f1     div      cx
000f:0000009e 8b56ee   mov      dx,word ptr [bp-12]
000f:0000009f 8946fc   mov      word ptr [bp-04],ax
000f:000000a0 f7e1     mul      cx
000f:000000a2 8946fc   mov      word ptr [bp-0c],ax
000f:000000a5 3946f6   cmp      word ptr [bp-0a],ax

U CS:IP-10 IP
000f:000000ae 3976f0   cmp      word ptr [bp-10],si
000f:000000b1 77df     ja       0092
000f:000000b3 3976f0   cmp      word ptr [bp-10],si
000f:000000b6 7510     jnz      00c8
000f:000000b8 c45ede   les      bx,word ptr [bp-22]
000f:000000bb 8b46f6   mov      ax,word ptr [bp-0a]
000f:000000be 268907   mov      word ptr es:[bx],ax

Observations About Unassembling From an Unknown Starting Place

Instructions are of variable length, from one to fifteen bytes long.
This means the address you provided may not actually be the start of an instruction. This also means, therefore, that the first few instructions you see may not actually be what the machine saw.

If you look at the output of several unassemblies starting at sequential addresses, you will see that after typically 3 to 5 tries, the unassembly will agree with previous ones, for some point after the unassembly started.

This is typically within four or five lines, but not always. Be cautious, and see if the sequence looks reasonable. If it does, you have most likely found an instruction boundary. Experience will help this process.

Some common sense will help as well. Obviously, an application in ring 3 cannot perform I/O directly. Likewise, the 'db' means that the unassembler did not have a way to interpret this as an instruction.

The last command entered looks at a few of the instructions which actually preceeded a failure. Can you discover which instruction put the data into the ES and BX registers?

Exercise 3: Unassembling and Reading Instructions

Objectives:
1. Reinforce the preceding lab exercises
2. Learn how to unassemble instructions
3. Learn how to read instructions
4. Learn about variable length instructions

We will now look at instructions.
1. In what type of segment are instructions found?
2. Are instructions EVER executed in any other segment type?
3. Unassemble the instructions which would have been next to execute (if the application hadn't trapped) by entering "U". The default address is CS:IP initially. You can unassemble further with repeated use of "U". To unassemble at a particular place, specify the address; for example CS:IP.
4. What was the next instruction which would have executed?
5. Unassemble using an address range to see some previous instructions. Type "U CS:IP-20 IP-10". This will unassemble from ip-20 to ip-10. Now type "U CS:IP-21 IP-10" and "U CS:IP-22 IP-10". Observe what is happening by closely observing the address at which each instruction begins.
6. Now type "U CS:IP-18 IP" to see the TWO instructions immediately before the failing instruction (at CS:IP!). What are they?
7. Which one loaded the address used in the next (failing) instruction?
8. Did the address come from this routine's private data, or was it a parameter passed by the caller?

This is presented in detail later.
9. Circumstantially at least, what seems to be wrong?

Also presented later.

This page left mostly blank so that the next pair of pages will face each other.

Exceptions
Events sometimes occur which disrupt the normal sequence of instruction. These are called exceptions and interrupts. Intel defines exceptions in relation to an unsuccessful attempt to execute an instruction. Interrupts are defined as a hardware response to a event unrelated to program execution.

<table>
<thead>
<tr>
<th>HEX</th>
<th>TYPE</th>
<th>B/C</th>
<th>ERR</th>
<th>SOURCE</th>
<th>CAUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Fault</td>
<td>C</td>
<td>No</td>
<td>Divide</td>
<td>Overflow (perhaps by zero)</td>
</tr>
<tr>
<td>1</td>
<td>DR6</td>
<td>B</td>
<td>No</td>
<td>Debug</td>
<td>Exception</td>
</tr>
<tr>
<td>2</td>
<td>Int</td>
<td>B</td>
<td>No</td>
<td>NMI</td>
<td>(Non-Maskable Interrupt), normally hardware fault</td>
</tr>
<tr>
<td>3</td>
<td>Trap</td>
<td>B</td>
<td>No</td>
<td>Breakpoint</td>
<td>(INT 3 instruction)</td>
</tr>
<tr>
<td>4</td>
<td>Trap</td>
<td>B</td>
<td>No</td>
<td>Overflow</td>
<td>(INTO instruction)</td>
</tr>
<tr>
<td>5</td>
<td>Fault</td>
<td>B</td>
<td>No</td>
<td>Bounds</td>
<td>Check (BOUND instruction)</td>
</tr>
<tr>
<td>6</td>
<td>Fault</td>
<td>B</td>
<td>No</td>
<td>Invalid</td>
<td>Opcode</td>
</tr>
<tr>
<td>7</td>
<td>Fault</td>
<td>B</td>
<td>No</td>
<td>Coprocessor not available, see note</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Abort</td>
<td>Always</td>
<td>Yes</td>
<td>Double</td>
<td>Fault, any instruction</td>
</tr>
<tr>
<td>9</td>
<td>Fault</td>
<td>C</td>
<td>Yes</td>
<td>Coprocessor Segment Overrun (286,386 only) (Fault D in 486+)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Fault</td>
<td>C</td>
<td>Yes</td>
<td>Invalid</td>
<td>TSS</td>
</tr>
<tr>
<td>B</td>
<td>Fault</td>
<td>C</td>
<td>Yes</td>
<td>Segment</td>
<td>Not Present (swapped out)</td>
</tr>
<tr>
<td>C</td>
<td>Fault</td>
<td>C</td>
<td>Yes</td>
<td>Stack</td>
<td>Exception</td>
</tr>
<tr>
<td>D</td>
<td>Fault</td>
<td>C</td>
<td>Yes</td>
<td>General</td>
<td>Protection</td>
</tr>
<tr>
<td>E</td>
<td>Fault</td>
<td>PF</td>
<td>Yes</td>
<td>Page</td>
<td>Fault (paged out)</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(reserved)</td>
</tr>
<tr>
<td>10</td>
<td>Fault</td>
<td>B</td>
<td>No</td>
<td>Coprocessor Error</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Fault</td>
<td>?</td>
<td>Always</td>
<td>Alignment</td>
<td>Check</td>
</tr>
<tr>
<td>12</td>
<td>Abort</td>
<td>??</td>
<td>Zero</td>
<td>Machine</td>
<td>Check</td>
</tr>
<tr>
<td>13-1F</td>
<td></td>
<td></td>
<td></td>
<td>(reserved)</td>
<td></td>
</tr>
<tr>
<td>20-FF</td>
<td>Trap</td>
<td>N/A</td>
<td>No</td>
<td>Available</td>
<td>for Hardware Interrupts Via 'INTR' Pin</td>
</tr>
<tr>
<td>00-FF</td>
<td>Trap</td>
<td>N/A</td>
<td>No</td>
<td>The INT instruction is actually a trap</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Co-processor not available may be due to not having one, or because the content of the co-processor belongs to another thread. The co-processor data needs to be saved and restored only when more than one thread is using it. Bit 3 in CR0 indicates that a thread switch has occurred and will cause a trap 7 when a co-processor instruction is decoded.

**Explanation of B/C column**
- B - Benign, means it is ok with any other exception
- C - Contributary, means it will contribute to a double fault
- PF - Page Fault, means a referenced address is not present

**Definition of Fault, Trap, Etc.**
1. Faults
CS & EIP point to the instruction which generated the fault.

2. Traps
CS & EIP point to the instruction to be executed after the instruction which caused the trap.
INT3, INTO, BOUND, and INT nn are examples of traps.

3. Aborts
In general, these exceptions do not permit locating the failing instruction, nor restart of the thread which caused the abort. Aborts are used to report inconsistent or illegal values in system tables, and hardware errors.

4. Interrupts
Unlike the preceding exceptions, interrupts are not related to the program being executed, but to an external condition.

--------------------------------------------

Hardware Error Codes

Selector Related Error Code
Bits 31-15: Reserved.
Bits 15-03: The index part of the selector involved.
Bit 02: The table indicator bit, if neither bit 01 nor bit 00 are 1.
Bit 01: IDT selector bit, if on, the selector is in the IDT.
Bit 00: External bit, if on, not caused by the program

Page Fault Error Code
Bits 31-04: Reserved.
Bit 03: RSV. A 1 bit was detected in a reserved bit of a page directory or page table entry.
Bit 02: U/S.
0: The program was in supervisor mode.
1: The program was in user mode.
Bit 01: W/R.
0: The access was a read.
1: The access was a write.
Bit 00: Level.
0: The fault is because of a not-present page.
1: The fault is because of page-level protection.

--------------------------------------------

Simultaneous Exceptions

It is possible for more than one exception to occur while attempting to execute an instruction. In order to determine what will happen if two simultaneous exceptions occur on the same instruction, use the following table:
<table>
<thead>
<tr>
<th>First</th>
<th>Second</th>
<th>Resulting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>Exception</td>
<td>Action</td>
</tr>
<tr>
<td>Benign</td>
<td>Benign</td>
<td>OK</td>
</tr>
<tr>
<td>Benign</td>
<td>Contributory</td>
<td>OK</td>
</tr>
<tr>
<td>Benign</td>
<td>Page Fault</td>
<td>OK</td>
</tr>
<tr>
<td>Contributory</td>
<td>Benign</td>
<td>OK</td>
</tr>
<tr>
<td>Contributory</td>
<td>Contributory</td>
<td>Double Fault</td>
</tr>
<tr>
<td>Contributory</td>
<td>Page Fault</td>
<td>OK</td>
</tr>
<tr>
<td>Page Fault</td>
<td>Benign</td>
<td>OK</td>
</tr>
<tr>
<td>Page Fault</td>
<td>Contributory</td>
<td>Double Fault</td>
</tr>
<tr>
<td>Page Fault</td>
<td>Page Fault</td>
<td>Double Fault</td>
</tr>
</tbody>
</table>

Note: OK means the faults are processed consecutively.

Note: Double Fault means the faults are reported together.

Note: If any other exception occurs trying to enter the DoubleFault handler, the processor shuts down until RESET, although, if the NMI handler has not been entered, NMI will be recognized and accepted.

Note: A trap C in Ring 0 is usually a double fault.

When the processor detects a Stack Exception it needs to push an error code and a return address onto the stack of the exception handler. If this happens in Ring 0, there will be no privilege level transition, which includes switching to a new, protected stack. If the exception is due to stack growth, there is no place to push the error code or return address.

RESULT: TRAP 8

The Address Space Picture

This is a picture of what the address space looks like for several processes.

```
FFFFFFFF -----------------------------------------------------
|                     KERNEL                          |
|            OS2 SYSTEM CONTROL BLOCKS                |
|                                                     |
|                   SYSTEM REGION                     |
|                                                     |
|                                                     |
|-----------------------------------------------------
|                                                     |
| 20000000  |                                                     |
|-----------------------------------------------------
| 1FFFFFFF  |                                                     |
|                                                     |
|                                                     |
|                                                     |
|                  SHARED REGION                      |
|                                                     |
| 13000000  |                                                     |
|-----------------------------------------------------
| 04000000  | 03000000  | 02000000  | 01000000  | 00000000  |
| PRIVATE REGION | PID 1 | PID 2 | PID 18 | PID 3 | PID 12 |
|               | EXPANSION REGION               | ( SHARED and PRIVATE ) |
```
Note: Within the private region you must know the Process ID, as well as the linear address to define a piece of virtual storage. All regions except the private region are shared among all processes. Above the private region in the shared regions, there is only one version of a given address, so you DO NOT need the Process ID.

Note: The boundary at 03FFFFFF is an initial value. If some application allocates over 03FFFFFF of private space, this boundary will move upward. It moves in steps of 00400000, because another page table is allocated.

Note: DLL's are initially loaded beginning at the 1BFFFFF boundary, and to successively lower addresses. This 'water mark' moves downward in steps of 00400000, too.

Note: Addresses not assigned to a memory object are invalid. Any attempt to use them will generate an exception.

Note: The address space picture discussed here is a simplified overview. A more detailed description may be found in the Advanced Guide to Hang Analysis chapter, under Memory Management and Ownership Topics.

OS/2 Implementation Details

This section discusses some of the implementation details of OS/2 which particularly involve debugging.

Shared Memory

This highlights how memory is shared among a few processes.

The same selector is allocated in each process that shares the storage. Each process therefore uses the same offset in the LDT, and the LDT entries are the same, so the same linear address is also used.

Note: The page table entries used for the shared storage are the same for both processes, too.

DLL's are a good example of shared storage.

DLLs are loaded into the shared address range. The boundary is dynamic, and moves downward as DLL's are loaded.

The boundary of private addresses move upward as private storage is allocated. There is a guarantee of 64 Meg for private, and 64 Meg for
Address Tiling

Address tiling refers to the practice of creating a mathematical or algorithmic relationship between an LDT selector and the base, or virtual address in the descriptor.

By using address tiling, OS2 avoids the need to move memory blocks because of reallocation, and also makes it very fast to convert an LDT Selector:Offset to a flat, or Linear Address. The implementation is simply to allocate 64K of virtual address space to each selector, starting with selector ‘000F’, at virtual address 64K, or ‘%100000’.

Note: Selector ‘0007’ is used to map the LDT as read-only data.

Why Thunk?

It is still common to have applications which have some 32-bit parts, and some 16-bit parts. The 32-bit parts try to avoid using 16-bit selector:offset addressing, because of the overhead of loading the selector registers, as well as to avoid the challenge of correctly dealing with storage references in both modes.

A typical example is a 32-bit application calling a 16-bit DLL.

Since storage is (must be) the same for all parts of a process, there has to be a way to convert one form of an address to the other.

Only 16-bit application selectors from the LDT are eligible for this quick form of the conversion, and only linear addresses less than %20000000 can be converted to 16:16 format.

Additionally, addresses in the packed region may NOT be converted by this quick method, but by a search of the LDT descriptor base (linear) addresses, followed by a calculation.

The top of normal application space, at %1BFFFFFF, is mapped to selector DFFF. The top of protected shared addresses at %1FFFFFFF maps to selector FFFF, if used.

Address Transformations (Thunks)

This section tells you how to change from 16:16 to 0:32-bit mode, or vice versa. There are two parts to thunking, the address transformation, and properly aligning the stack, if necessary. The stack alignment is usually done by a subroutine which detects the need to do this, and builds an ‘extra’ frame in the new mode, properly aligned by making a copy of the incoming parms, transforming the addresses as part of this process.

This works only because the specific implementation within OS/2 which was designed to use address tiling for LDT selectors.

16:16 to 0:32 Thunk

The selectors which are eligible for this thunk are LDT selectors which are PL=3. In this case, all three low-order bits are 1. Because of this, one can shift the selector three bits to the right, or divide by 8, without loss of information. The resulting number is the high-order word of the 32-bit address because of address tiling. For example, address 000F:008A can be thunked from 16:16 to 0:32 as follows:

```
0 0 0 0 F : 0 0 B A <--- Hex Sel:Offset
0000 0000 0000 1111 0000 0000 1011 1010 <--- Binary
shift the selector 3 bits to the right, which gives
```
A stack may require alignment, because a 32-bit stack is built on double-word boundaries, with two low order zero bits in the address of each element, whereas a 16-bit stack is aligned only on a word boundary.

0:32 to 16:16 Thunk

Because the range of LDT selectors is only 512 Meg, addresses less than this can be transformed to use an LDT selector, with restrictions. The transformation is to append three low-order 1 bits to the value, and to discard three high order zero bits. An alternative way of stating this is to multiply by 8, then add 7. The three low order one bits are LDT (table indicator=1), and PL=3. The restrictions are that the storage must be PL=3 application storage, must not span a 64K boundary in the linear address space, and the value must be less than hex 2000 0000.

Simultaneous 16-bit and 32-bit Descriptions of Virtual Storage

---
Note: The digits within the tables are the offsets to each descriptor. The selector values (CS=0F) indicate which selector normally accesses the descriptor.

Note: Any selector containing the value 0-3 is the NULL selector which DOES NOT specify the first entry in the GDT. It is a place holder when a selector does not specify a descriptor. Any attempt to use the null selector results in a general protection exception.

Note: The descriptors in the LDT are 16-bit descriptors. This is one of the reasons that 16-bit programs still execute and fail in exactly the same manner as on previous versions of OS/2.

Stacks

This section describes how most OS/2 programs use the stack.

Understanding the stack is generally straightforward. The stack is defined by the descriptor selected by the Stack Selector register or SS, and the stack pointer or SP. Stacks are always read/write. There are two basic operations on a stack, PUSH and POP. PUSH decrements the stack pointer and then stores the operand at the offset provided by SP in the stack segment. POP moves the data item at the offset provided by SP to the operand and then increments SP. SP ALWAYS POINTS TO THE LAST ITEM PUSHED. Stacks grow downward from higher addresses to lower addresses.

Near CALL & RETurn

The near CALL instruction is used to invoke a subroutine. The instruction first pushes IP into the stack and then updates IP so that it contains the offset of the first instruction in the subroutine.

The near form of the RETurn instruction is really just a POP IP, which restores the saved content of IP. Execution continues at the instruction following the CALL.

Far CALL & RETurn

The far CALL instruction is used to invoke a subroutine. The instruction first pushes CS into the stack, and then pushes IP. Next, it updates CS & IP so that they contain the selector:offset of the first instruction in the subroutine.

The far form of the RETurn instruction first pops IP, which restores the saved content of IP, and then pops CS, restoring it as well. Execution continues at the instruction following the CALL.

Passing Parameters

Parameters are generally passed to a subroutine by putting them on the stack with PUSH instructions prior to the CALL. Parameters are removed from the stack in one of two ways:

- By the caller ("C" convention), generally adding a constant to SP.
- By the subroutine (PASCAL convention), by specifying the operand for the RETurn which is added to SP after the return address is POP'ed.

Note: "C" convention PUSHes parameters from right to left.

Note: PASCAL convention PUSHes parameters left to right.

Because the NEAR versions of jump (JMP), CALL and RETurn DO NOT touch CS, there can be no change of privilege level during execution of any of them. The FAR versions of them do provide a new value for CS. If the new CS is the same privilege level as the current
Receiving Parameters

There is a register which can be used by a subroutine to access parameters very efficiently. This register is the Base Pointer. When it is used to obtain an offset, the default segment is the STACK SEGMENT. If the entry to a subroutine begins with these instructions the stack will look like the picture on the next page.

```
PUSH        BP
MOV         BP,SP
SUB         SP,sizeof( LOCAL DATA ITEMS )
```

This sequence is so common that there is a single instruction equivalent:

```
ENTER       sizeof( LOCAL DATA ITEMS ), 0
```

This allows all parameters to be accessed as BP plus the appropriate offset and local data elements to be accessed as BP minus the appropriate offset.

The instructions to exit are either:

```
MOV         SP,BP
POP          BP
RET          
```
or:

```
LEAVE
RET
```

Why Do We Care About the Pascal Convention?

The Pascal convention was used by OS/2 1.x for those calls which access system functions which are implemented in a higher privilege level (ring) than the application. It is also used to call 16 bit Window Procedures. Two examples are DosAllocSeg and DosRead. The decision was made to use the Pascal convention because of the way the hardware protects access to instructions and storage which is more privileged. This type of interface, including hardware operation, is discussed in detail after basic stack operation has been discussed.

Single Stack Frame
Note: A stack grows downward (expand down).

When this convention is followed the stack can be viewed as a series of 'stack frames'. Each stack frame has parameters and local data for some routine and linkage to the 'stack frames' used by the caller of that routine, etc. The saved BP values create a linked list in the stack segment which has all the information about each call including the return address. The process of following the chain back is referred to as 'unwinding the stack' and is an important aid to diagnosis when working on a problem.

--------------------------------------------

An Example of Using the Stack

This is a trivial example of how to pass and receive parameters, which is used to document where the stack pointer and base pointer are at the end of each instruction.

The example is 32-bit non-optimized code.

The subroutine, SUB, is designed to return the difference obtained by subtracting the second parameter from the first.

First, the relevant C code:

```c
( main )                 ( sub )
..                    ..
z=sub(A,B);                int sub(int x, int y)
..                    .
..                    .
..                    .
```

Next, the assembler code

```
( i ) initial condition
PUSH B     ; (01)             SUB:    PUSH EBP         ; (04)
PUSH A     ; (02)                     MOV  EBP,ESP     ; (05)
CALL SUB   ; (03)                     SUB  ESP,nn      ; (06)
ADD  ESP,8 ; (12)                      .
MOV  Z,EAX ; ( f ) final condition     .   ( NOTE )
  ..                      .
MOV  EAX,[EBP+8] ; (07)
  ..                      .
MOV  ESP,EBP       (09)
POP  EBP           (10)
RET                (11)
```

Note: At this point, the stack frame is established. If another, lower-level routine is called, the code to do so will look like the code seen in main, and a new stack frame will be established by that routine as soon as it receives control.

The new frame will be just below the current one.

--------------------------------------------

Stack Example

This example shows a stack, with ESP on the left, and EBP on the right.

<table>
<thead>
<tr>
<th>ESP</th>
<th>EBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ higher addresses }</td>
<td>{ i,10 }</td>
</tr>
</tbody>
</table>
Note: The numbers in parentheses indicate where the register points immediately after the numbered instruction on the previous page completes.

Multiple Stack Frames

STACK SEGMENT
---------------------------- high addresses

<table>
<thead>
<tr>
<th>high addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

---PARMS FROM ASTART---
---RETURN TO ASTART----
ASTART'S BASE POINTER-
----------------------------

---PARMS FROM MAIN-----
---RETURN TO MAIN------
---MAIN's BASE POINTER-
----------------------------

---PARMS FROM SUB 1----
---RETURN TO SUB 1-----
---SUB 1 BASE POINTER--
----------------------------

---PARMS FROM SUB 2----
---RETURN TO SUB 2-----
---SUB 2 BASE POINTER--
----------------------------

BP --> SUB 2 BASE POINTER--
----------------------------

SP --> SUB 3 FRAME

----------------------------

A Stack From a Dump
The first word, which is addressed by the current value in the BP register, is the near address of the next stack frame, 1550.

The next two words are a far return address, with the offset to the left of the selector. The return is to address F:F1.

The words following the return address are the parameters, if any were passed. There is no direct way to tell from the stack how many parameters were passed, or expected. To see the next frame,

In this stack frame, the BP chain pointer is zero. This usually means that you have found all of the frames on this stack.

The return address for this frame is F:300. The parameters seem to be an integer, 1, and three far addresses, 1F:1560, 1F:156e, and 1F:1568. A little further inspection shows that the third address 1F:1568 is pointed to by the first, which is highly unusual. Actually, this is the stack frame received by 'main'.

Main's parameters are as follows:

1. an integer, which tells it how many strings were found on the command line
2. the far address of a list of addresses, each of which points to one of the strings
3. the far address of a second list of addresses, each of which points to an environment variable. This list is terminated with a NULL POINTER, a far address in which both the selector and offset are zero.

Let's look at them.
1F:1560 has the address 1f:1568. Near addresses default to the last selector used, so we are not required to supply it every time.

Right, the name of the program was the first string on the command line. The first parameter indicates that there is only one string.

Let's look at a few of the environment variables.

This gets us four far addresses. To see them all with only one input line, use the semicolon as a command delimiter and type away.

Notice that the tools are not very particular about spaces in the commands.

Lastly, to see the local data for the failing routine,

and now you have it, displayed above.
Application Documentation

We will briefly discuss what files are optionally generated by most compilers, and how to tell the linker to create the map file. After an explanation of the contents, and why some of the numbers are what they are, we will answer some questions using various parts of the optional application documentation.

The .MAP File

When you look at a 16-bit map file, you will discover that it may have at least three sections. A 32-bit map file can have at least 4.

1. The first section is built in the same sequence as the executable.
2. The second section contains a list of all external symbols, sorted by the name of the symbol. This is particularly useful when a programmer wants to find where some particular variable or routine is located.
3. The third section contains a list of the same symbols, sorted by the location of the symbol. This is particularly useful when you know where something is, and want to find out if it has an external name, or what routine encompasses the address of interest.
4. The fourth section of a 32-bit map file contains a list of locations where the compiled code for each input line begins. This can tell you almost immediately which line of code failed, once you know which program, and where within the program the failing instruction was located.

The .COD File

Many 16-bit compilers will produce a file similar to a .COD file, although it may have a different file extension. For example, MicroFocus Cobol can produce a .GRP file, which has the same organization as the .COD file.

The format of this file is that of a mixed listing.

The listing generally contains an input line, identified by line number, followed by the machine instructions generated by the compiler, with the address to the left, the hex instruction in the middle, and an assembler form of the instruction on the right. Some of these files will actually be accepted as is by an assembler, but most compilers document the fact that this is not a supported feature of the compiler.

Obviously, if you know the offset of some instruction, perhaps one that caused a failure, you can use this listing to identify which line of the input program caused the generation of the failing instruction.

Exercise 4: Application Documentation

Some typical files associated with 16- and 32-bit applications

A 16-bit Map File
Part 1: Same sequence as executable.

```
DEMO

<table>
<thead>
<tr>
<th>Start</th>
<th>Length</th>
<th>Name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001:0000</td>
<td>00292H</td>
<td>DEMO_TEXT</td>
<td>CODE</td>
</tr>
<tr>
<td>0001:0292</td>
<td>02BE6H</td>
<td>_TEXT</td>
<td>CODE</td>
</tr>
<tr>
<td>0001:2E78</td>
<td>00000H</td>
<td>C_ETEXT</td>
<td>ENDCODE</td>
</tr>
<tr>
<td>0002:0000</td>
<td>02910H</td>
<td>FAR_BSS</td>
<td>PAR_BSS</td>
</tr>
<tr>
<td>0003:0000</td>
<td>00042H</td>
<td>NULL</td>
<td>BEGDATA</td>
</tr>
<tr>
<td>0003:0042</td>
<td>007D8H</td>
<td>_DATA</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:081A</td>
<td>00000H</td>
<td>XIFB</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00000H</td>
<td>XIFE</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00000H</td>
<td>XIB</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00000H</td>
<td>XI</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00000H</td>
<td>XIE</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00000H</td>
<td>XPB</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:0828</td>
<td>00004H</td>
<td>XP</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XPE</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XCB</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XC</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XCE</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XCFB</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XCF</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00000H</td>
<td>XCFE</td>
<td>DATA</td>
</tr>
<tr>
<td>0003:082C</td>
<td>00006H</td>
<td>CONST</td>
<td>CONST</td>
</tr>
<tr>
<td>0003:0832</td>
<td>0000BH</td>
<td>HDR</td>
<td>MSG</td>
</tr>
<tr>
<td>0003:083A</td>
<td>000FAH</td>
<td>MSG</td>
<td>MSG</td>
</tr>
<tr>
<td>0003:0934</td>
<td>00002H</td>
<td>PAD</td>
<td>MSG</td>
</tr>
<tr>
<td>0003:0936</td>
<td>00001H</td>
<td>EPAD</td>
<td>MSG</td>
</tr>
<tr>
<td>0003:0938</td>
<td>00226H</td>
<td>__BSS</td>
<td>BSS</td>
</tr>
<tr>
<td>0003:0B5E</td>
<td>00000H</td>
<td>XOB</td>
<td>BSS</td>
</tr>
<tr>
<td>0003:0B5E</td>
<td>00000H</td>
<td>XO</td>
<td>BSS</td>
</tr>
<tr>
<td>0003:0B5E</td>
<td>00000H</td>
<td>XOE</td>
<td>BSS</td>
</tr>
<tr>
<td>0003:0B60</td>
<td>00000H</td>
<td>c_common</td>
<td>BSS</td>
</tr>
<tr>
<td>0003:0B60</td>
<td>00A00H</td>
<td>STACK</td>
<td>STACK</td>
</tr>
</tbody>
</table>

Origin  Group
0003:0  DGROUP
```

Note: The numbers to the left of the colon look like the selector part of a far address, because that is what they will become. The linker has no idea what selectors will be assigned by the loader, so it simply calls the first segment 1, the next segment 2, and so on.

Note: The loader actually builds a table that shows the relationship between the selector assigned and the segment number from the map.

Part 2: Sorted by the name of the symbol

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Publics by Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:0000</td>
<td>Imp DOSALLOCSEG (DOSCALLS.34)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSCHGFILEPTR (DOSCALLS.58)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSEXIT (DOSCALLS.5)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETDBCSEV (NLS.4)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETMACHINEMODE (DOSCALLS.49)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETVERSION (DOSCALLS.92)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSHANDTYPE (DOSCALLS.77)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSREAD (DOSCALLS.137)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSREALLOCSEG (DOSCALLS.36)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSSETVEC (DOSCALLS.89)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSWRITE (DOSCALLS.138)</td>
</tr>
<tr>
<td>0003:06E6</td>
<td>STKHOQ</td>
</tr>
<tr>
<td>0001:2D3E</td>
<td>_brkctl</td>
</tr>
<tr>
<td>0003:0938</td>
<td>_edata</td>
</tr>
<tr>
<td>0003:0B60</td>
<td>_end</td>
</tr>
<tr>
<td>0003:069B</td>
<td>_environ</td>
</tr>
<tr>
<td>0003:0662</td>
<td>_errno</td>
</tr>
<tr>
<td>0001:057A</td>
<td>_exit</td>
</tr>
<tr>
<td>0001:24E6</td>
<td>_fflush</td>
</tr>
<tr>
<td>0001:03F0</td>
<td>_fgets</td>
</tr>
<tr>
<td>0001:295C</td>
<td>_flusshall</td>
</tr>
<tr>
<td>0001:275C</td>
<td>_free</td>
</tr>
<tr>
<td>0001:0000</td>
<td>_gen</td>
</tr>
</tbody>
</table>
```
Part 3: Sorted by location in storage

<table>
<thead>
<tr>
<th>Address</th>
<th>Publics by Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETMACHINEMODE (DOSCALLS.49)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETVERSION (DOSCALLS.92)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSREAD (DOSCALLS.137)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp __AHINCR (DOSCALLS.136)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSTEXIT (DOSCALLS.5)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSALLOCSEG (DOSCALLS.34)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSREALLOCSEG (DOSCALLS.38)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSCHGFILEPTR (DOSCALLS.58)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSWRITE (DOSCALLS.138)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSSETVEC (DOSCALLS.89)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSQHANDTYPE (DOSCALLS.77)</td>
</tr>
<tr>
<td>0000:0000</td>
<td>Imp DOSGETDBCSEV (NLS.4)</td>
</tr>
<tr>
<td>0000:9876</td>
<td>Abs __acrtmsg</td>
</tr>
<tr>
<td>0000:9876</td>
<td>Abs __acrtused</td>
</tr>
<tr>
<td>0000:6D06</td>
<td>Abs __aDbdoswp</td>
</tr>
<tr>
<td>0000:6D06</td>
<td>Abs __aDbused</td>
</tr>
<tr>
<td>0001:0000</td>
<td>_gen</td>
</tr>
<tr>
<td>0001:00E2</td>
<td>__main</td>
</tr>
<tr>
<td>0001:02A2</td>
<td>__astart</td>
</tr>
<tr>
<td>0001:0306</td>
<td>__cintDIV</td>
</tr>
<tr>
<td>0001:0310</td>
<td>__amsg_exit</td>
</tr>
<tr>
<td>0001:0394</td>
<td>__printf</td>
</tr>
<tr>
<td>0001:03F0</td>
<td>__fgets</td>
</tr>
<tr>
<td>0001:0492</td>
<td>__sscanf</td>
</tr>
<tr>
<td>0001:04F0</td>
<td>__cinit</td>
</tr>
<tr>
<td>0001:057A</td>
<td>__exit</td>
</tr>
<tr>
<td>0001:0591</td>
<td></td>
</tr>
</tbody>
</table>
A 16-Bit Code File

;       Static Name Aliases
;       $S180_inbuf     EQU     inbuf
TITLE   DEMO.C
.286p
.287
DEMO_TEXT       SEGMENT  WORD PUBLIC 'CODE'
DEMO_TEXT       ENDS
_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS
CONST   SEGMENT  WORD PUBLIC 'CONST'
CONST   ENDS
_BSS    SEGMENT  WORD PUBLIC 'BSS'
_BSS    ENDS
DGROUP  GROUP   CONST, _BSS, _DATA
ASSUME  CS: DEMO_TEXT, DS: DGROUP, SS: DGROUP
EXTRN   __acrtused:ABS
EXTRN   _printf:FAR
EXTRN   _sscanf:FAR
EXTRN   _fgets:FAR
_BSS      SEGMENT
_COMM NEAR       _prime:  2:      5000
_BSS      ENDS
_BSS      SEGMENT
$S180_inbuf     DW 028H DUP (?)
_BSS      ENDS
CONST      SEGMENT
$T20004 DW SEG _prime

Program entry point at 0001:02A2
CONST ENDS
DEMO_TEXT SEGMENT
ASSUME CS: DEMO_TEXT
***
*** #include <stdio.h>
***
*** #define INBUFSIZE 80
*** #define NPRIME 5000
*** int gen(void);
***
***
PUBLIC _gen
_gen PROC FAR
*** 000000 c8 24 00 00 enter WORD PTR 36,0
*** 000004 57 push di
*** 000005 56 push si
***
ix = -6
l = -16
ll = -14
q = -4
t = -10
tp = -8
tt = -12
*** unsigned short ix,l=2,ll=25,npr=3,q,t,tp=2,tt;
***
*** 000006 c7 46 f0 02 00 mov WORD PTR [bp-16],2
*** 00000b c7 46 f2 19 00 mov WORD PTR [bp-14],25
*** 000010 c7 46 fe 03 00 mov WORD PTR [bp-2],3
*** 000015 c7 46 f8 02 00 mov WORD PTR [bp-8],2
***
prime[0]=2;
***
*** 00001a 8e 06 00 00 mov ax,6
*** 00001f 2b 46 f8 sub al,WORD PTR [bp-8] ;tp
***
prime[1]=3;
***
*** 000025 26 c7 06 02 00 03 00 mov WORD PTR es:_prime+2,3
***
prime[2]=5;
***
*** 00002c 26 c7 06 04 00 05 00 mov WORD PTR es:_prime+4,5
***
for ( t=7 ; t<65530 ; t+=tp )
***
*** 000033 c7 46 f6 07 00 mov WORD PTR [bp-10],7
*** 000038 c7 46 e2 04 00 mov WORD PTR [bp-30],OFFSET _prime+4
*** 00003d c7 46 e4 00 00 mov WORD PTR [bp-28],SEG _prime
*** 000042 c7 46 e6 00 00 mov WORD PTR [bp-34],OFFSET _prime+6
*** 000047 c7 46 e0 00 00 mov WORD PTR [bp-32],SEG _prime
$L20002:
***
***
***
*** 00004c b8 06 00 mov ax,6
*** 00004f 2b 46 f8 sub ax,WORD PTR [bp-8] ;tp
*** 000052 89 46 f8 mov WORD PTR [bp-8],ax ;tp
***
*** if { ll<=t }.
***
*** 000055 b8 46 f6 mov ax,WORD PTR [bp-10] ;t
*** 000058 39 46 f2 cmp WORD PTR [bp-14],ax ;ll
*** 00005b 77 15 ja $L170
***
***
*** 1++;
***
*** 00005d 83 46 e2 02 add WORD PTR [bp-30],2
*** 000061 ff 46 f0 inc WORD PTR [bp-16] ;l
***
ll=prime[1]*prime[1];
***
*** 000064 c4 5e e2 les bx,WORD PTR [bp-30]
*** 000067 26 8b 07 mov ax,WORD PTR es:[bx]
for ( ix=2 ; ix<l ; ix++ )

q = t / \text{prime}[ix];

if ( t =\text{tt} ) break;

\text{if} ( l = ix ) \text{prime}[npr++] = t;

\text{return} npr;
PUBLIC _main

_main PROC FAR
"
*** 0000e2 c8 60 00 00 enter WORD PTR 96,0
*** 0000e6 57 push di
*** 0000e7 56 push si;
; argc = 6
; argv = 8
; ix = -6
; last = -10
; nf = -8
; fact = -76
; input = -2
; is = -12
; q = -4
;*** static char inbuf[INBUFSIZE];
;*** int ix,last,nf;
;*** unsigned short fact[32],input=0,is,q;
; Line 36
*** 0000e8 c7 46 fe 00 00 mov WORD PTR [bp-2],0 ;input
;
;*** last=gen();
; Line 37
*** 0000ed 0e push cs
*** 0000ee e8 00 00 00 call NEAR PTR _gen
*** 0000f1 c7 46 f6 00 00 mov WORD PTR [bp-10],ax ;last
;*** printf("there are %u primes less than 65536\n",last);
; Line 38
*** 0000f4 50 push ax
*** 0000f5 push ds
*** 0000f6 68 00 00 push OFFSET DGROUP:$SG188
*** 0000f9 9a 00 00 00 00 call FAR PTR _printf
*** 0000fe 83 c4 06 add sp,6
;*** if ( 1<argc )
; Line 39
*** 000101 83 7e 06 01 cmp WORD PTR [bp+6],1 ;argc
*** 000105 7e 25 jle $I190
;***     if ( 0==sscanf(argv[1],"%u",&input) ) argc=1;
; Line 40
*** 000107 8d 46 fe lea ax,WORD PTR [bp-2] ;input
*** 000109 80 00 push ss
*** 00010b 50 00 push ax
*** 00010c le 0e push ds
*** 00010d 68 25 00 push OFFSET DGROUP:$SG191
*** 000110 c4 5e 08 les bx,DWORD PTR [bp+8] ;argv
*** 000113 26 ff 77 06 push WORD PTR es:[bx+6]
*** 000117 26 ff 77 04 push WORD PTR es:[bx+4]
*** 00011b 9a 00 00 00 00 call FAR PTR _sscanf
*** 000120 83 c4 0c add sp,12
*** 000123 0b c0 or ax,ax
*** 000125 75 05 jne $I190
*** 000127 c7 46 fe 01 00 mov WORD PTR [bp+6],1 ;argc
;*** while ( 2>argc )
; Line 41
$I190:
*** 00012c 83 7e 06 02 cmp WORD PTR [bp+6],2 ;argc
*** 000130 7d 4b jge $FB194
*** 000132 8b 76 06 mov si,WORD PTR [bp+6] ;argc
$L20005:
;*** }
; Line 42
;*** printf("Enter number to factor: ");
; Line 43
*** 000135 1e push ds
*** 000136 68 28 00 push OFFSET DGROUP:$SG195
*** 000139 9a 00 00 00 00 call FAR PTR _printf
*** 00013e 83 c4 04 add sp,4
;*** fgets(inbuf,INBUFSIZE,stdin);
; Line 44
*** 000141 1e push ds
*** 000142 68 00 00 push OFFSET __iob
*** 000145 6a 50 push 80
*** 000147 1e push ds
*** 000148 68 00 00 push OFFSET DGROUP:$SG180_inbuf
*** 00014b 9a 00 00 00 00 call FAR PTR _fgets
*** 000150 83 c4 0a add sp,10
if (0 == sscanf(inbuf, "%u", &input))
{
    lea    ax, WORD PTR [bp-2]; //input
    push   ss
    push   ax
    push   ds
    push   OFFSET DGROUP:$SG197
    push   OFFSET DGROUP:$SI180_inbuf
    call   FAR PTR _sscanf
    add    sp,12
    or      ax, ax
    jne    $FB194
    printf("Unable to convert number. Please try again\n");
    push   ds
    push   OFFSET DGROUP:$SG198
    call   FAR PTR _printf
    add    sp,4
}
else break;
}
for (ix=nf=0, is=input ; ix<last ; ix++)
{
    sub    ax, ax
    mov    WORD PTR [bp-8], ax; //nf
    mov    WORD PTR [bp-6], ax; //ix
    mov    WORD PTR [bp-2]; //input
    cmp    WORD PTR [bp-10], 0; //last
    jg     $JCC399
    shl    ax, 1
    add    ax, OFFSET _prime
    mov    WORD PTR [bp-90], ax
    mov    WORD PTR [bp-88], SEG _prime
    lea    ax, WORD PTR [bp-76]; //fact
    mov    WORD PTR [bp-94], ax
    mov    WORD PTR [bp-92], ss
    mov    WORD PTR [bp-10]; //last
    sub    ax, WORD PTR [bp-6]; //ix
    add    WORD PTR [bp-6], ax; //ix
    mov    WORD PTR [bp-8], ax; //ix
    xor    ax, ax
    jc     $FB205
    sub    ax, WORD PTR [bp-6]; //ix
    sub    dx, dx
    div    WORD PTR [bp-86]
    add    WORD PTR [bp-6], ax; //ix
    mov    ecx, WORD PTR [bp-2]; //input
}
q = input / prime[ix];
while (q*prime[ix] == input)
{
    les    bx, DWORD PTR [bp-90]
    mov    ax, WORD PTR es:[bx]
    mov    WORD PTR [bp-86], ax
    mov    ax, cx
    sub    dx, dx
    div    WORD PTR [bp-86]
    mov    WORD PTR [bp-4], ax; //q
**Line 53**
```assembly
{  
    fact[nf++]=prime[ix];
}
```

**Line 54**
```assembly
*** 000f0 26 89 35  mov  WORD PTR es:[di],si
*** 001f3 83 c7 02  add  di,2
*** 001f6 ff 46 f8  inc  WORD PTR [bp-8]; nf
```

**Line 55**
```assembly
input/=prime[ix];
```

**Line 56**
```assembly
*** 00205 2b d2  sub  dx,dx
*** 00207 f7 f6  div  si
*** 00209 89 46 fc  mov  WORD PTR [bp-4],ax ; q
```

**Line 57**
```assembly
*** 002c 8b 46 b2  mov  ax,WORD PTR [bp-78]
*** 002f 7d 14  jge  $I206
*** 0028 77 66 fc  mul  WORD PTR [bp-4]; q
*** 0022 3b c1  cmp  ax, cx
*** 0024 74 da  je  $L20006

FB205:
```

**Line 58**
```assembly
*** 00222 83 7e f8 02  cmp  WORD PTR [bp-8],2 ; nf
*** 00224 7d 14  jge  $I206
*** 00226 ff 76 f4  push  WORD PTR [bp-12]; jis
*** 00228 e8 89 fa 01 00  add  WORD PTR [bp-6], ix
*** 0022a 77 0a 00 00 00 push  OFFSET DGROUP:$SG207
*** 0022c 9a 00 00 00 00  call  FAR PTR _printf
*** 0022e 83 c4 06  add  sp,6
*** 00233 5e  pop  si
*** 00235 5f  pop  di
*** 00237 cb  leave
*** 00239 90  nop
$I206:
```

**Line 59**
```assembly
*** 0023b ff 76 b4  push  WORD PTR [bp-76]; fact
*** 0023d ff 76 f4  push  WORD PTR [bp-12]; jis
*** 00241 4e  dec  si
*** 00243 83 c4 08  add  sp,8
```

**Line 60**
```assembly
*** 00245 83 c4 01 00  add  WORD PTR [bp-6], si ; ix
*** 00247 7e 29  jle  $FB211
*** 00249 8d 46 b6  lea  ax,WORD PTR [bp-74]
*** 0024b 89 46 a2  mov  WORD PTR [bp-94],ax
*** 00250 ff fe f8  mov  WORD PTR [bp-92], mm
*** 00252 8b 76 f8  mov  si,WORD PTR [bp-8]
*** 00255 4e  dec  si
*** 00256 01 76 fa  add  WORD PTR [bp-6], si ; ix
$L20010:
```

**Line 61**
```assembly
*** 00258 83 c7 e8 01  cmp  WORD PTR [bp-8],1 ; nf
*** 0025a 7e 29  jle  $FB211
*** 0025c 8d 46 b6  lea  ax,WORD PTR [bp-74]
*** 0025e 89 46 a2  mov  WORD PTR [bp-94],ax
```

**Line 62**
```assembly
*** 00260 01 76 fa  add  WORD PTR [bp-6], si ; ix
```
Questions

Please answer the following questions, using the preceeding listings:

1. How large is segment 2 of DEMO.EXE? ____________
2. What is the segment:offset of the 'fgets' routine? ____________
3. What is the segment:offset of the symbol 'fpinit'? ____________
4. Does DEMO.EXE call DosOpen? ______ How can you tell? ____________
5. Which routine begins at address 0001:29DE? ____________
6. How long is the routine named 'strlen'? ____________
7. Which routine contains address 0001:186A? ____________
8. How far into the routine is the previous address? ____________
9. What is the program's entry point? ____________
10. What is the name of the routine which is the entry point? ____________
11. What instruction mnemonic is at offset 00C5 in DEMO.EXE? ____________
12. What variable is in AX when the return at 00E0 executes? ____________
13. Which line in DEMO.C generated the above return? ____________
14. Offset 0188 in DEMO.C is in which C function? ____________
15. What variable name is used by the instruction at 0055? ____________
16. What is the purpose of the instruction at offset 0234? ____________

Note: In the .cod file, the numbers in the assembler instructions are decimal.

The lines generated in the .cod file between offsets 00E7 and 00E8 Tell you where the local variables are stored, relatively speaking.
17. To what are the numbers like 8, -10, -76, -12 relative? _____________

18. If a failure were to occur in routine 'gen', what command would you use to display only the variable 'npr'?   DW _____________

19. How would you display the variable 'Y'?   DW _____________

20. Is the variable 'tp' in 'gen' at the same location as the variable 'nf' in main?   Yes / No Explain. ________________________________________________________________________________

21. Check the offsets for 'gen' and 'main' between the .map and the .cod files. Do they match? _______ Why/why not? __________ Will the offsets always behave this way?   Yes / No Explain. ________________________________________________________________________________

A 32-bit Map File

<table>
<thead>
<tr>
<th>Start</th>
<th>Length</th>
<th>Name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001:00000000</td>
<td>000001A68H</td>
<td>CODE32</td>
<td>CODE 32-bit</td>
</tr>
<tr>
<td>0001:00000300</td>
<td>00000006CH</td>
<td>CODE4_DATA32</td>
<td>DATA 32-bit</td>
</tr>
<tr>
<td>0003:00000000</td>
<td>00000005CH</td>
<td>DATA32</td>
<td>DATA 32-bit</td>
</tr>
<tr>
<td>0003:0000005C</td>
<td>0000000B0H</td>
<td>CONST32</td>
<td>CONST 32-bit</td>
</tr>
<tr>
<td>0003:0000010C</td>
<td>000000000H</td>
<td>BSS32</td>
<td>BSS 32-bit</td>
</tr>
<tr>
<td>0003:00000110</td>
<td>000002000H</td>
<td>STACK32</td>
<td>STACK 32-bit</td>
</tr>
</tbody>
</table>

Origin | Group  
0000:0  | FLAT  
0003:0  | DGROUP

Address | Publics by Name
0000:00000000 | Imp DOS32FLATTOSEL (DOSCALLS.425)
0001:000001A7A | DOS32GETMESSAGE
0001:000001A7A | Dos32GetMessage
0000:00000000 | Imp DOS32QUERYMESSAGEC (MSG.8)
0000:00000000 | Imp DOS32SELTOFLAT (DOSCALLS.426)
0000:00000000 | Imp DOS32TRUEGETMESSAGE (MSG.6)
0000:00000000 | Imp DosAllocMem (DOSCALLS.299)
0000:00000000 | Imp DosExit (DOSCALLS.234)
0000:00000000 | Imp DosFreeMem (DOSCALLS.304)
0001:000001A7A | DosGetMessage
0001:000001A7A | DOSGETMESSAGE
0000:00000000 | Imp DosWrite (DOSCALLS.282)
0001:00000F94 | free
0001:00000000 | gen
0001:00000000 | main
0001:000001A4 | malloc
0001:000001A8 | sig32
0002:00000000 | __argc
0002:00000000 | __argv
0001:000001A4 | _bufprint
0001:000001850 | _DosFlatToSel
0001:000001848 | _DosSelToFlat
0003:0000010C | _edata
0001:00000F48 | _edcGetMessage
0003:00000110 | _end
0002:00000000 | __exeentry
0002:0000010D | _have_free
0001:0000010D | _heapmin
0001:00001108 | _heapmin_int
0001:00001108 | _ilog2
0001:00001108 | _log2
0001:00001108 | _Nsynmsg"
### Address Publics by Value

<table>
<thead>
<tr>
<th>Segment</th>
<th>Offset</th>
<th>Import Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:00000000</td>
<td>Imp DosFreeMem</td>
<td>(DOSCALLS.304)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DOS32QUERYMESSAGEEGCE</td>
<td>(MSG.8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DOS32SELOTOFLAT</td>
<td>(DOSCALLS.426)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DosAllocMem</td>
<td>(DOSCALLS.299)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DOS32TRUEGETMESSAGE</td>
<td>(MSG.6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DOS32FLATTOSEL</td>
<td>(DOSCALLS.425)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DosWrite</td>
<td>(DOSCALLS.282)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>Imp DosExit</td>
<td>(DOSCALLS.234)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001:00000000</td>
<td>gen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>main</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__printf_ansi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__printfieee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__bufprint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__dofmto</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_edcGetMessage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>free</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_heapmin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_heapmin_int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_ilog2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__isdigit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>malloc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_MesgServ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_PrintErrMsg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_DosSelToFlat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_DosFlatToSel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_setuparg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_terminate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_wfloatfmt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>sig32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_exeentry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__argc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__argv</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_have_freed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>__bucketArr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_edatas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000:00000000</td>
<td>_end</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Line numbers for DEMO.obj(DEMO.C) segment CODE32

<table>
<thead>
<tr>
<th>Source Line Num</th>
<th>Src File Index</th>
<th>Flags (OX)</th>
<th>Seg:Offset (OX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>00</td>
<td>0001:00000000</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>00</td>
<td>0001:00000009</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>00</td>
<td>0001:0000000e</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>00</td>
<td>0001:00000024</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>00</td>
<td>0001:0000002b</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>00</td>
<td>0001:00000032</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>00</td>
<td>0001:00000040</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>00</td>
<td>0001:0000004b</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>00</td>
<td>0001:00000050</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>00</td>
<td>0001:00000051</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>00</td>
<td>0001:0000005a</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>00</td>
<td>0001:00000069</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>00</td>
<td>0001:00000082</td>
</tr>
</tbody>
</table>
A 32-bit .ASM File, Produced by CSET/2
public gen

    proc
    push ebp
    mov ebp, esp
    push ebx
    push esi
    push edi
    mov [ebp+8], eax; prime
    mov DWORD PTR [ebp-18], 019h;      ll
    mov DWORD PTR [ebp-14], 03h;       npr
    mov DWORD PTR [ebp-10], 02h;       tp
    Prime[0]=2;
    mov DWORD PTR [EAX], 02h
    Prime[1]=3;
    mov DWORD PTR [EAX+4], 03h
    Prime[2]=5;
    mov DWORD PTR [EAX+08], 05h
    for ( t=7; t<65530; t+=tp )
        mov ecx, [ebp-1ch]; ix
        mov ebx, 07h
    for ( ix=2; ix<l; ix++ )
        mov ECX, [EBP-01ch]; ix
        mov edi, 02h

FELB6:

    mov edx, [EBP-00H]; tp
    neg edx
    add edx, 06h
    mov [ebp-10], edx; tp
    if ( ll<=t )
        l++;
        ll=prime[l]*prime[l];
        mov edx, DWORD PTR [EAX+edi*4];
        imul edx, edx
        mov [ebp-18], edx; ll
    FELB7:

    mov ecx, 02h
    cmp edi, 02h
    jle FELB8
    mov edx, DWORD PTR [EAX+edi*4];
    imul edx, edx
    mov [ebp-18], edx; ll
    FELB7:

    for ( ix=2; ix<l; ix++ )
        mov ecx, 02h
        cmp edi, 02h
        jle FELB8

FELB9:

    mov [EBP-020H], edi; @CBE17
    mov esi, eax
    g=t/prime[ix];
    tt=q*prime[ix];
    if ( t==tt ) break;
    mov edi, DWORD PTR [esi+ecx*04h];
    mov eax, ebx
    cdq
    idiv edi
    mov edx, edi
    mov edi, [EBP-020H]; @CBE17
    xchg esi, eax
    imul edx, esi
    cmp edx, ebx
    je FELB8

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:

    mov ecx, 02h
    inc ecx

FELB7:

    mov ecx, 02h
    inc ecx

FELB8:
CMP ECX, EDI
JL FELB9

FELB8:

;;;; 29  if ( l==ix ) prime[npr++]=t;
CMP EDI, ECX
JNE FELB12
MOV EDX, [EBP-014H]; npr
MOV DWORD PTR [EAX+EDX*04H], EBX
INC EDX
MOV [EBP-014H], EDX; npr
FELB12:
MOV EDX, EBX

;;;; 30  }
MOV EBX, [EBP-010H]; tp
ADD EBX, EDX
CMP EBX, 0fffaH
JL FELB6

;;;; 31  return npr;
MOV EAX, [EBP-014H]; npr
POP EDI
POP ESI
POP EBX
LEAVE
RET
gen ENDP

;;;; 34  int main(int argc, char *argv[])
ALIGN 04H
PUBLIC main
main PROC
PUSH EBX
PUSH ESI
PUSH EDI
SUB ESP, 0cH

;;;; 39  rc=DosAllocMem(&mem,16384,PAG_READ+PAG_WRITE+PAG_COMMIT);
PUSH 013H
PUSH 04000H
LEA ECX, [ESP+010H]; mem
PUSH ECX
MOV AL, 03H
CALL DosAllocMem
ADD ESP, 0cH

;;;; 40  if ( rc ) return printf("non-zero return code from DosAllocMem=\d\n",rc);
OR EAX, EAX
JE FELB18
PUSH EAX
MOV EAX, OFFSET FLAT: @STAT1
SUB ESP, 04H
CALL __printfIEEE
ADD ESP, 014H
POP EDI
POP ESI
POP EBX
RET
FELB18:

;;;; 41  last=gen(p=mem);
MOV EAX, [ESP+08H];  mem
MOV [ESP+04H], EBX;  p
CALL gen
MOV ESI, EAX

;;;; 42  printf("there are %u primes less than 65536\n",last);
PUSH ESI
MOV EAX, OFFSET FLAT: @STAT2
SUB ESP, 04H
CALL __printfIEEE
MOV EAX, ESI
ADD ESP, 08H
Questions

Please answer the following questions, using the preceding listings:

1. How large is segment 1 of DEMO.EXE? ___________
2. What is the segment:offset of the 'bufprint' routine? ___________
3. What is the segment:offset of the symbol 'have_freed'? ___________
4. Does DEMO.EXE call DosWrite? _____ How can you tell? ___________
5. Which routine begins at address 0001:12E8? ___________
6. How long is the routine named 'terminate'? ___________
7. Which routine contains address 0001:1888? ___________
8. What is the program's entry point? ___________
9. What is the name of the routine which has the entry? _____________

10. How far into this routine is the actual entry point? _____________

11. What is the first instruction mnemonic generated by line 28? _____________

12. What variable is in EAX when the return at line 31 executes? _____________

13. Offset 0124 in DEMO.C is in which C function? _____________

14. What variable name is used by the instruction at 0040? _____________

15. Where does the code for line 34 start? _____________

   Note: In the .asm file, the numbers in the assembler instructions are hex. You can tell because they are suffixed with 'H'.

   The assembler code generated has the variable name following each line where it is referenced. This makes it easy to locate the variables, because you simply use the address expression in the instruction.

16. Look at line 11. To what are the numbers -18, -14, -10 relative? _____________

17. Look at the code generated for line 15. Where will the variable 't' be found? _____________

18. If a failure were to occur in routine 'gen', what command would you use to display only the variable 'npr'? DD _____________

19. How would you display the variable 't'? DD _____________

20. Is the variable 'l' in 'gen' at the same location as the variable 'ix' in main? Yes / No Explain. _____________

--------------------------------------------

Exercise 5: Unwinding a 16-bit Stack

Objectives:

1. To learn how to 'unwind' a stack. This is how to find the calling hierarchy which existed at some particular point, such as at the point of failure.

2. To learn how to 'mine' information from the stack frames.

Normally, every routine which has not returned to its caller will have a stack frame. Each stack frame normally contains the parameters passed to a routine, the return address for the routine, and the data which is local for that routine.

Start the dump formatter just as before, on the same dump.

Questions:

1. The convention states that BP or EBP will point to the current stack frame. SP will point to the lowest address which is in use. Therefore, note the initial values for SP _______ and BP _______. Since SS is the selector that defines the stack, note which it is. Some analysts also note the limit of the SS descriptor, because that value bounds the range of both SP and BP.

   SS _______ SSLIM _______

2. Display the current stack frame using DW SS:BP. This will show you only part of the frame, but most analysts do this because it makes following the chain so easy.

   The first word is the offset, or near address, of the next frame. The second word is the offset part of the return address. If the call was a far call, the return must also be a far call. If this is the case, the third word is the selector part of the return address.
next stack frame ________ return offset ________ selector ________

3. Some number of words following the return address are the parameters passed. There is no way to know for certain how many parameters there are, unless you know how both the caller and the routine are written. Analysts typically write down a few words, as convenient.

parameter word# 1 ________ 2 ________ 3 ________ 4 ________

4. At this point we have gleaned what we can from this frame. Now you need to repeat the process for the rest of the stack frames.

Many analysts will follow the entire chain of stack frames before going to the system or application documentation to find the names of the routines involved, and the line numbers. Others choose to go back and forth, and put in the routine names and line numbers for each frame as they go.

The application documentation will tell you where variables are stored. Remember that each routine uses its own stack frame, so be certain to use the numeric value rather than the register name 'BP' to look at local data for routines other than the failing one.

If you display from SP to BP-2, or ESP to EBP-4, you will see the entire local data for the routine using the current stack frame. This can be quite nice for locating the individual variables.

Find the routine which failed by looking at the .MAP file.

Find the line number that failed by looking next at the .COD file.

The following variables are involved in the failure: 'npr' and 't'. their locations can be found in the .COD file.

Find the location of npr,________ then display its value ________

Find the location of t,________ then display its value ________

You may want to look at the call to the failing routine, before going away to find the programmer.

Exercise 6: Unwinding a 32-bit Stack

Objectives:

1. To learn how to 'unwind' a stack. This is how to find the calling hierarchy which existed at the point of failure.

2. To learn how to 'mine' information from the stack frames.

Normally, every routine which has not returned to its caller will have a stack frame. Each stack frame normally contains the parameters passed to a routine, the return address for the routine, and the data which is local for that routine.

Start the dump formatter by typing DF_RET ..\DUMPS.162\DUMP04.DMP

Questions:

1. The convention states that BP or EBP will point to the current stack frame. ESP will point to the lowest address which is in use.

Therefore, note the initial values for ESP _______ and EBP _______. Since SS is the selector that defines the stack, note which it is.

SS _______ SSLIM _______ (not generally useful when SS is 53)

2. Display the current stack frame using DD SS:EBP. This will show you only part of the frame, but most analysts do this because it makes following the chain so easy.

The first doubleword is the offset, or near address, of the next frame. The second doubleword is the offset part of the return address. If the call was a far call, the return must also be a far call. If this is the case, the third doubleword is the selector part of the return address.

IT IS RARE FOR 32-BIT PROGRAMS TO USE FAR ADDRESSES.

next stack frame ________ return offset ________

3. Some number of doublewords following the return address are the parameters passed. There is no way to know for certain how many parameters there are, unless you know how both the caller and the routine are written. Analysts typically write down a few doublewords, as convenient.
4. At this point we have gleaned what we can from this frame. Now you need to repeat the process for the rest of the stack frames.

Many analysts will follow the entire chain of stack frames before going to the system or application documentation to find the names of the routines involved, and the line numbers. Others choose to go back and forth, and put in the routine names and line numbers for each frame as they go.
The application documentation will tell you where variables are stored. Remember that each routine uses its own stack frame, so be certain to use the numeric value rather than the register name 'BP' to look at local data for routines other than the failing one.

If you display from ESP to EBP-2, or ESP to EBP-4, you will see the entire local data for the routine using the current stack frame. This can be quite nice for locating the individual variables.

Find the routine which failed by looking at the ·MAP file.

Find the line number that failed by looking again at the ·MAP file.

The following variables are involved in the failure: 'npr' and 't'. their locations can be found in the ·ASM file.

Find the location of npr,________ then display its value ________

Find the location of t,________ then display its value ________ Hint: t has been optimized, and is in a register.

You may want to look at the call to the failing routine, before going away to find the programmer.

--------------------------------------------

Requesting Kernel Services

If CALL targets a less privileged CS, or RET ( RETURN ) a more privileged CS, a general protection exception occurs by definition; a trusted program cannot directly invoke a less trusted one.

If CALL targets a more privileged CS, a general protection exception occurs because a less privileged program cannot access a more privileged object (code segment).

It is IMPOSSIBLE to DIRECTLY call a code segment which is a different privilege level than the caller.

It IS POSSIBLE to INDIRECTLY call a more privileged code segment.

--------------------------------------------

The Task State Segment (TSS)

This hardware control block is used to control hardware multitasking, I/O access, and privilege transitions.

--------------------------------------------

How to Find the TSS

There is a selector register named the task register (TR). This register has a GDT selector that chooses a descriptor whose type is TSS. This descriptor contains the base and limit for the TSS.

Task State Segment Format

The fields from offset 4 to 1F are not changed by the hardware.

<table>
<thead>
<tr>
<th>Offset (size)</th>
<th>Content</th>
<th>Offset (size)</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>00(2)</td>
<td>link - previous tss selector</td>
<td>08(2)</td>
<td>Ring 0 SS</td>
</tr>
<tr>
<td>04(4)</td>
<td>Ring 0 ESP</td>
<td>10(2)</td>
<td>Ring 1 SS</td>
</tr>
<tr>
<td>0C(4)</td>
<td>Ring 1 ESP</td>
<td>18(2)</td>
<td>Ring 2 SS</td>
</tr>
<tr>
<td>14(4)</td>
<td>Ring 2 ESP</td>
<td>20(4)</td>
<td>EIP</td>
</tr>
<tr>
<td>1C(4)</td>
<td>CR3.</td>
<td>28(4)</td>
<td>EAX</td>
</tr>
<tr>
<td>24(4)</td>
<td>EFLAGS</td>
<td>30(4)</td>
<td>EDX</td>
</tr>
</tbody>
</table>
The Call Gate

An explanation of what a call gate provides, and how it works.

Why Have a Call Gate?

The CALL GATE is the mechanism by which an application requests services from the operating system. Integrity has several requirements which are not immediately obvious to most people.

1. The caller must be forced to use a designed entry point to prevent entry at an arbitrary location; for example, at a point after the parameters have been validated. This might cause the operating system to violate its own integrity or that of another application.

2. The parameters, as well as the rest of the stack, must be protected from the application while in use by the operating system to prevent changes by another thread in that application.

3. The return address must be protected from the application while the operating system is running to prevent other threads of the application from altering it in a way that would cause a return to the application in a privileged mode.

Note: A CALL GATE implements all of the above requirements.

Note: A CALL GATE is a system descriptor which describes an entry point in a more privileged program which is accessible to less privileged programs.

Another View

A Gate is a ‘service window’ which describes the entry point of the gate and what size package is passed into the more protected ring.
Call Gate Contents

<table>
<thead>
<tr>
<th>CALL GATE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PL of Gate</td>
<td>Can I see this gate?</td>
</tr>
<tr>
<td>CS of entry</td>
<td>Where is the entry?</td>
</tr>
<tr>
<td>EIP of entry</td>
<td>Where is the entry?</td>
</tr>
<tr>
<td>Parm Count</td>
<td>What gets passed?</td>
</tr>
<tr>
<td>WC or DWC</td>
<td></td>
</tr>
<tr>
<td>A Descriptor</td>
<td></td>
</tr>
</tbody>
</table>

Note: Observe that the privilege level of the gate controls which privilege level programs can access the gate; the target privilege level is contained in the entry point CS value.

Call Gate Overview

When a FAR CALL contains a target code selector (CS) which is a CALL GATE, the processor ignores the offset (IP) contained in the instruction and gets the true target CS and offset (IP) from the CALL GATE. In addition, if the call is to a more privileged program, the processor locates a fresh stack for it to use, stores the current stack selector and stack pointer in the new, more privileged stack, copies the parameters from the old stack to the new one, and finally saves the return information in the new stack. All this happens during the execution of the call instruction.

Briefly, when the return occurs, all this gets undone.

Call Gate Detail

From less to more privileged, for example, Ring 3 to Ring 0

1. Verify new stack will hold linkage data. If not, stack fault, error code 0.
2. New SS, SP from TSS, based on PL of new CS.
3. Old SS:SP copied to new stack.
4. Parameters (up to 15 words or doublewords) copied from old to new stack.
5. Former CS:IP copied to new stack; SP now points here.
6. New CS, IP from Call Gate

A Ring Transition Picture
The following is how the stacks look at entry to the more privileged program:

<table>
<thead>
<tr>
<th>RING 3 STACK</th>
<th>RING 0 STACK &lt;-- INITIAL R0 ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>high addresses</td>
<td>SS</td>
</tr>
<tr>
<td>P3</td>
<td>ESP</td>
</tr>
<tr>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td>P1</td>
<td>P2</td>
</tr>
<tr>
<td>P0</td>
<td>P1</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>low addresses</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td>EIP</td>
</tr>
</tbody>
</table>

**Note:** There is NO RETURN ADDRESS on the less privileged stack.

**Note:** The two items at the top of the more privileged stack are the less privileged SS and ESP.

**Note:** Subtract 8 from the SP value found in the TSS to find where the less privileged ESP and SS are stored. The values in the TSS are initial values, not the address of the first item pushed.

**Note:** A trap C in Ring 0 is usually a double fault.

When the processor detects a Stack Exception it needs to push an error code and a return address onto the stack of the exception handler. If this happens in Ring 0, there will be no privilege level transition, which includes switching to a new, protected stack. If the exception is due to stack growth, there is no place to push the error code or return address.

RESULT: TRAP 8

--------------------------------------------

Return Detail

From more to less privileged, for example, Ring 0 to Ring 3

1. Verify that all steps below will work. If not, general protection fault.
2. Pop IP, CS
3. Add immediate value to 'old' SP
4. Pop SP, SS
5. Add immediate value to 'new' SP
6. Zero every selector which has PL more privileged than the new CPL. This is required to maintain integrity because access validation is done only when a selector register is changed - not when it is used.

--------------------------------------------

Exercise 7: Looking at a Ring Transition

Objectives:

1. Introduction
To review previous knowledge of analysing traps
To begin getting familiar with the debug kernel
To learn how to identify the API targeted by a call gate

2. Techniques
To learn about the PATCH program
To learn about getting control when a module is loaded

3. Finding the TSS and the privileged stacks
To learn when you may need to find the TSS
To learn how to find the TSS
To learn how to find privileged stacks

4. Watching a ring transition
Look at the ring 0 stack before
Look at the ring 3 stack before
Actually execute a far call from ring 3 to ring 0.
Look at both stacks afterwards.

Part 1: Introduction to the Debug Kernel

Procedures: Introduction
1. Change to directory CLASS/LABS/LAB09
2. Execute OSPREY, see the failure, and the trap screen.
   At the failure, record CS:EIP from the trap screen.
   CS _________     EIP _____________
At this point, it is too late to cause a dump. Dismiss the trap screen.

We will refer to the system on which the problem occurs at the Machine Under Test, or the MUT. The MUT is connected via a null modem cable to an adjacent machine, which we will call the debug terminal. Most of the debugging actions will occur from the debug terminal, on which we will run a public domain terminal emulation program, LOGICOMM. If you like LOGICOMM and intend to use it frequently, you should register it, which will also get you an improved version.

Let’s use the debug kernel for the first time. First, we need to get its attention. The way to do this is to enter Control-C on the debug terminal, after starting LOGICOMM. The debug kernel defaults to settings 9600, N, 8, 1
3. Start LOGICOMM on the debug terminal, then type Control-C.
   The default response of the debug kernel is the registers at whatever point OS/2 was interrupted by the Control-C. This is not generally very useful. We need to get control where we want it, not at a random place.
4. Enter the command VSF*
   This tells the debug kernel that you want control on any interrupt which may be Fatal to a thread. The ‘F’ is for fatal, the ‘*’ is for ‘any’.
   Enter the command G (Go), so OS/2 can continue.
5. **On the MUT, rerun OSPREY.**

This time, you should get a group of lines on the debug terminal which tell you that a fatal failure has occurred.

Enter the command `DG CS` You will find that this is in ring 0.

Before we look at ring 0, let us find where ring 3 called ring 0, and also identify the API which was called.

Enter the command `.R` *(the period is very significant!)*

`.R` shows you the ring 3 registers, whereas `R` shows you the current ones.

```
CS=__________   EIP=__________   Does this match the trap screen?
```

```
eax=00000000  ebx=0000405c  ecx=00000000  edx=00000001  esi=000016b0  edi=00000000
```

We already know this instruction did not trap; the trap is in ring 0.

6. **If we unassemble prior to 000f:1bc3, we will find a far call.**

```
...1BBE   call   ___________:0000
```

The instruction as hex data is _____ _____ _____ _____ _____

7. If you inquire about the descriptor by entering `DG` and the selector, You should see something similar to this

```
# DG lxxx
lxxx CallG32 Sel:Off=0148:0000550a  DPL=3 P DW=7
```

Write down CS ______   EIP__________   DPL________   DWC________

If you enter the `LN` command with the values of CS and EIP from the call gate, you will identify the API which is called via this gate.

8. **We know how to find parameters on the ring 3 stack, DW SS:SP**

We can also find them on the ring 0 stack, but at this point, the kernel has already manipulated some of the addresses, so there is not an exact match. We need to get control at the point of the call at 1BBE.

9. **Enter the command `GT` which will GoThrough the trap.**

---

**Part 2: Some Techniques**

Procedures to get control at a point other than a trap:

1. **We will make use of a couple of breakpoint commands**

   This command tells the debug kernel that we want control on the debug terminal at some specific point. The problem is that the place where we would like to get control is not loaded into memory until we run the program, and it is difficult at best to type Control-C at just the right time.

   **Enter the command `BP DosLibIDisp,".p*"`**

   The content of the quoted string is the command to execute when we arrive at the breakpoint. This will assure us that we are in the correct context, because the output of `.p` includes the module name.

   Let the MUT run, and execute OSPREY once again.
You will probably get control in the context of OSPREY. If not, issue 'g' again a time or two until you are.

3. At this point, OSPREY has been loaded, so we can set a breakpoint.

If you simply try the command BP 0F:1BBE, you will discover that the page is not yet loaded. There are two ways around this problem.

a. Use a register breakpoint, BR E:0F:1BBE

b. Cause OS/2 to bring the page in with .I 0F:1BBE

Then reenter the BP command from above.

4. However, this is ‘cheating’ because we already knew where to stop.

To find the address of the first instruction at this point, enter the command .M 0F:0

Find the MTE handle, hmte.

Issue the .LMO command with the HMTE as the parameter.

Alternatively, try .LMO 'OSPREY', which works sometimes.

The output of the .LMO command includes the linear address of the MTE.

Display the MTE as doublewords, and get the address of the SMTE from the output; it is in the second doubleword.

Display the SMTE as doublewords, and you can find the entry point in the second and third words displayed.

Now you can set a breakpoint at the entry to any module.

The PATCH program

1. On the MUT, execute the EXEHDR utility against OSPREY.EXE.

EXEHDR is distributed with the developers’ toolkit.

The output will provide you information you need to patch a program successfully. The last part of the output should look like

<table>
<thead>
<tr>
<th>Module:</th>
<th>OSPREY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>OSPREY.EXE</td>
</tr>
<tr>
<td>Data:</td>
<td>NONSHARED</td>
</tr>
<tr>
<td>Initial CS:IP:</td>
<td>seg 1 offset 0088</td>
</tr>
<tr>
<td>Initial SS:SP:</td>
<td>seg 3 offset 0000</td>
</tr>
<tr>
<td>Extra stack allocation:</td>
<td>0a00 bytes</td>
</tr>
<tr>
<td>DGROUP:</td>
<td>seg 3</td>
</tr>
</tbody>
</table>

There are two things we will need in this listing.

2. The entry point, or initial CS:IP is __________:_________

3. The location in the file where that segment begins __________

The columns labelled 'file' and 'mem' are the sizes of the segment in the file, and in memory. The difference is due to uninitialized data, which is not stored, saving space and reducing program load time.

To find the location of an instruction in the file, add the offset to the file address.

4. To get control, we will replace a byte with the hex value ‘CC’, which is a special one-byte instruction, Int 3, or BreakPoint.

5. We will patch the call instruction at 1BBE.

Add the offset, 1BBE to the file address 0200 __________

If you cannot add hex, get the debug kernel’s attention, and then type in ? 1BBE+0200. ? is a general purpose evaluation command.

6. We now know where we want to patch the program. Let’s do it.

On the MUT, enter the command PATCH OSPREY.EXE
The patch address was calculated above; enter it.

The byte you are about to replace is hex ______

Type CC then press enter, and complete the confirmations.

We have now patched the program.

7. Execute the program on the MUT; you get control at the INT 3.

We need to put back the hex data which was originally there, so as not to introduce another problem. We will use the enter command.

Type the command E CS:IP

You will see the 'CC', type the original data value and press enter.

Type the command .R and you should see the original far call.

8. This is one way to get control.

It has problems if the MUT is not where you can touch it.

Type the commands G then GT to let OSPREY finish.

9. Patch OSPREY back to its original content if you wish.

Part 3: Finding the TSS

It is relatively simple to find and display this critical control block which is used by the hardware for ring transitions.

1. Get the debug kernel's attention, so you can display data.

2. The TSS is located via the Task Register (TR), which is a selector.

You can find the value in TR by entering ? TR

Entering RT toggles register terse mode. Try R before and after entering RT. You can look for TR in the output.

You really do not want TR, but the TSS, which is at TR:0.

3. Enter

DD TR:0 to display the TSS as doublewords

DT TR:0 to format the TSS.

4. The first doubleword is the link field.

It indicates which TSS called this one through a task gate.

The next two doublewords are the ESP and SS for entry to ring 0.

The next pair of doublewords are unused by OS/2; they would have the ESP and SS for entry into ring 1.

The next pair of doublewords are the ESP and SS for entry to ring 2.

5. To display the stack used at entry to ring 0,

use the DD command with the SS and ESP values from the TSS; BUT Stacks grow downward, so put -80 after the ESP value. 80 is the number of bytes displayed by default; this will show you the top of the stack for ring 0, with the saved SS value as the last item shown.

Part 4: Watching a Ring Transition
We will watch a ring transition by stopping on an instruction which we know causes a ring transition, display both stacks, then single step the instruction, and look at both stacks again.

Get control in OSPREY so that the next instruction is at 0F:1BBE.

1. Display the call gate by using DG and the selector from the call.
   Write down the target CS______ EIP_________ DWC_____ PL____

2. Display the ring 3 stack as WORDS
   so you can see as many DOUBLEWORDS as are passed through the gate.
   Display the ring 0 stack as words, too. It is technically incorrect to do this, but for the purposes of this exercise, it makes things easy.

3. Use the command T to execute the call instruction.

Now, again display the ring 0 stack as words again.

4. Compare the ring 0 content now with the content of the ring 3 stack.

Do not overlook the ring 0 SS and ESP at the top of the ring 0 stack.

Do not overlook the return address in the ring 0 stack, following the parameters which were copied by the hardware as it executed the call.

5. TIMESAVER:
   If you know what API will be called, you can simply set the breakpoint at the API, by using its name. A side effect is that every thread which calls the API will stop, so you may want to use something like ’p*’ as the command to execute at the breakpoint, which makes it easy to see when the thread of interest is there.

   This lab is now complete. However, if you let it run to the failing instruction, you will find an additional detail about this API, namely that because only 13 words were pushed, and 7 doublewords are needed to get them all copied into the ring 0 stack, there is one more detail we can see, namely how the difference (two bytes) is handled.

   If you display the ring 0 stack once again, it has been changed!

   The return will need to add enough to the ring 0 stack pointer that it can find the ring 3 stack successfully; this is also what is added to the ring 3 stack pointer, because both stacks must be cleaned up. In order that this not be destructive of what is already on the ring 3 stack, the ring 0 entry code has adjusted the saved ring 3 ESP downward by 2 before the trap occurs. This is an example of some of the work that has been done within the ring 0 stack by the privileged code.

Exercise 8: Identifying the Owner of Storage

Objectives:

1. To learn how to find out where a part of storage originated
2. To learn how to find out what module contains it, if not dynamically acquired

Every piece of storage has an owner. Storage owned by OS/2 may not have all the storage accounting information which is kept for storage used by applications. The most common clue that this situation has occurred is the presence of the 'UVirt' flag (bit 52) in the descriptor. The next most common clue is that the procedure below may fail if complete storage accounting has not been done.

Within OS/2, handles are used extensively. Generally, a handle is nothing more than an index into some table or other. For diagnostic purposes, one can treat it as a 'magic number' that can be used as an operand on certain commands.

The initial objective is to find the module table entry which the loader built when the module was loaded. This will relate storage to the 'far' addresses in the link map.

The procedure is slightly different for private and shared storage.

With practice, one learns quickly what selectors are likely to be private, and which are likely to be shared. Refer to the address space picture which appears earlier, to refresh your memory about private and shared storage.
One way to tell is to display the entire LDT (using 'DL'), and to look for the gap between 'low numbered' and 'high numbered' selectors.

If the address is private, there will likely be many processes that define the address, and the data is likely different for each. You will need to find which process is the one you want.

The dump formatter command '.I' will show you not only the handle of the module table entry for the executable which caused this process to exist, but also will show you the handle of the 'PTDA', which is the key control block for a process.

The command usually used to identify storage is the '.M' command.

If issued with a shared address, the output has the handle of the module table entry. If issued for a private address, you get a set of output lines for every process which contains the address. In this case, you will need to use the hPTDA, or PTDA handle from the '.I' command to determine which set of output lines to use.

Once you have the handle of the module table entry, issue the command .LMO <handle>.

The command will not only give you the full path name of the module, but will also format a table which has a column (toward the right) titled 'sel'. This is the selector assigned. The first line of output is for the first segment in the link map, the second line is for the second segment, and so on. Thus, you can convert the selector:offset in the dump to a segment:offset in the correct link map.

Start the dump formatter by typing DF_RET ..\DUMPS.162\DUMP01.DMP

Procedure:
1. Enter the command '.I'

The PDTA handle is _______, the module table entry handle is _______.
2. Enter the command '.M CS:IP' to identify Memory at CS:IP.

Which 'har' line is for our process? har=_________

What is the hmte value from this set of lines? hmte=______________

Note: This is exactly what the '.I' command showed you, because this is what the .I command does internally.

3. Enter the command '.LMO ', followed by the hmte value.

What is the full path name of the module that contains CS:IP?

4. What is the segment number which has been assigned selector 000F? ______

5. What address would you look for in the link map to find CS:IP?

6. Now, repeat the same steps using the data in the next few displays. The address of interest is DFDF:9324

7. What is the privilege level of this segment? ______

8. What is its size? ___________

9. What is the command to identify memory at this address? __________________

Issue it. The lines which start hco= are context records, which indicate all of the contexts (processes) that can reference this address. It is extremely likely to be a shared address.

10. Issue the '.LMO' command for the module table entry handle.

11. What module is this? Full path name is ______________________

12. Which segment in the module contains this address? _________

13. Therefore, in the .map file, the address will be ______:_______

Steps to Diagnose a Trap
The intent of the following material is to illustrate a proven method for finding the cause of a trap in an application program. By first learning how to solve the simplest problems, one will have a much better basis for approaching more difficult problems. Historically, problem solving skills have been largely self-taught. Much can be learned by observing others solve problems. Many problems can be solved quickly by using significant short-cuts and assumptions and then verifying them. When a novice observes an experienced diagnostician, the activities are difficult to understand, and may lead to the opinion that each problem has its own special method for solution, which in turn leads to questions about when to use which method.

The following process will lead you to the cause of a trap.

Remember to take notes as you proceed. This will help if you are interrupted, and want to continue later, or if you need to explain to someone else what you found, and what facts led you to a particular analysis of the situation. You can obviously do this manually, but you can use a log file more easily. Just type ‘?’ followed by whatever you wish to log. The tools will evaluate the string, supplying the trailing quote, and show you the string, thus adding your thoughts to the log.

1. Locate the failing instruction.
   If you cannot do this, you have no place to start. Most operating systems will provide at least an excellent clue to the location of the failing instruction, if not its exact address.

2. Determine why the failing instruction will not execute.
   A knowledge of hardware operation, or a reference manual kept handy, is essential for this step. At the very worst, each of the possible exceptions described in the manual can be eliminated one by one until the cause is found.

Until you know why the instruction will not execute, you do not know what went wrong at the machine level. Conversely, as soon as you do know, you are prepared to begin the diagnosis of how things got into such a state. Observe that this does not require knowledge of C, FORTRAN, COBOL, SMALLTALK, etc. It requires only hardware knowledge.

3. Analyse how the conditions for failure occurred. It may be that an address calculation was done incorrectly, or that the failure was due to an invalid parameter. If the former, you now need only to discover what program has done this, and where in that program the error occurred. Skip the next two steps.

4. If an invalid parameter has been received, you must now update your notion of the cause of the problem. You need to consider the call as the location of the failure, and the specific parameter value as the reason why the called routine did not execute.

5. You must now analyse how the parameter was created, and where it came from. Unwind one stack frame, and return to step 3.

6. You now know what caused the problem, and now it is time to identify the failing program, locate the failing line, find the value of the program's variables, and, in general, collect all the data the programmer would have had if the failure had occurred at his desk. This step is usually a mechanical one.

Once this is done, go find the programmer, and turn over all you know about the problem. Be prepared to continue helping, or to show the programmer how to get additional information.

--------------------------------------------

The OS/2 System Trace

The System Trace facility is used to record a sequence of system events, function calls, or data in a fixed-size circular buffer as requested by calls to its API's. The buffer must be allocated during the processing of CONFIG.SYS.

If you have a TRACEBUF statement in CONFIG.SYS, a trace buffer is allocated, allowing you to use the TRACE command successfully later.

If any valid TRACE statements are in CONFIG.SYS (including TRACE=OFF), a trace buffer will be allocated for the default size of 4K, if TRACEBUF is not specified. This means that the statement TRACE=OFF will actually enable system tracing, which seems counter-intuitive to many people.

If you do not specify TRACE or TRACEBUF statements in CONFIG.SYS, OS/2 does not allocate a trace buffer and system tracing is disabled.

After the trace data is recorded, the trace formatter is used to retrieve the data from the system trace buffer and present it on your display, and optionally, to print it, or save it in a file.

--------------------------------------------
TRACEBUF and TRACEFMT

TRACEBUF=x

TRACEBUF sets the size of the trace buffer in the CONFIG.SYS file.

The parameter ‘x’ specifies a circular trace buffer size of up to 63K. If you have a TRACEBUF statement without a TRACE statement in CONFIG.SYS, the trace buffer size requested is specified and tracing is turned off (the same as if you specify TRACE=OFF).

If you need to use the System Trace facility, use the largest size, if possible. TRACEBUF=63

TRACEFMT displays formatted trace records in reverse time-stamp order. It is intended to be used to format the trace data so that you can analyse the content of the trace buffer. The most recent entry is displayed first. TRACEFMT numbers each event as it is formatted. The event numbers are unrelated to the trace data, and are useful when discussing a trace with someone else, for easy reference to events.

TRACEFMT works without a filename only if you have a trace buffer defined in the running system.

TRACEFMT works with a filename only if the file is a hex image of a trace buffer from a system for which you have Trace Formatting Files. If the .TFF file is not correct, the entries which are different will be formatted incorrectly with little or no indication of an error.

The file is typically created by the dump formatter by using the command ".TS filename", but TRACEFMT will also save the trace buffer in unformatted form. This is much smaller than the formatted form.

--------------------------------------------

TRACE and TRACE Processing

The trace command is used to control the system trace.

Command line: CONFIG.SYS:

TRACE ON TRACE=ON
TRACE OFF TRACE=OFF
(you can specify only static TRACE in the CONFIG.SYS file).

The above is optionally followed by one or more major code specifications, or one or more trace definition file specifications, or keywords. Next, you may optionally specify one or more process identifiers. Finally, you may specify that the trace buffer be cleared, and that trace activity be suspended, or resumed.

OS/2 processes TRACE statements in the order in which they appear from any source. TRACE commands in CONFIG.SYS are processed in the order they appear. The effects of the statements are cumulative for the duration of OS/2’s execution. If any part of a statement is incorrect, OS/2 ignores the statement.

Process Id is specified by /P:nn,nn,nn (where nn is in HEX!)

Clearing the trace buffer is specified by using /C.

Resuming trace activity is specified by using /R.

Suspending trace activity is specified by using /S.

Major and minor event codes are associated with all trace events. Some of the major codes follow:

Machine Exceptions Major Code: 3
Hardware Interrupts Major Code: 4
Device Helper Routines Major Code: 6
Disk Device Driver Major Code: 7

Major codes may be specified by listing them separated by commas, or as a range, for example, 2-7 specifies codes 2,3,4,5,6,7. Both methods may be combined, as in 5,7,12-18,27-32,9.
If you do not specify TRACE in CONFIG.SYS, event tracing is not started by CONFIG.SYS processing, but may be started later if TRACEBUF has been specified.

Records in the buffer are identified by major and minor codes. Some of the data that may be recorded in the circular buffer includes system events such as interrupts, exceptions, and thread switches.

OS/2 contains a mixture of static tracepoints and dynamic tracepoints.

Static tracepoints are implemented as trace function calls within individual software modules. The TRACE command can be used to turn on and off static tracepoints by specifying them by major code and, optionally, by minor code.

Dynamic tracepoints are implemented by implanting an INT 3 instruction at the specified location, and gathering data when the interrupt occurs. The TRACE command can be used to turn on and off dynamic tracepoints, but only by specifying the module or trace definition file name as a parameter. Dynamic tracepoints cannot be turned on and off by reference to their major codes.

TRACEFMT Processing

When the trace is complete, you can use the trace formatter (TRACEFMT) to format the data into a report. This helps you to isolate causes of problems in the OS/2 system by making the data in the trace buffer available for analysis.

Static & Dynamic Trace, and Files Used

Trace Format Files (.TFF) & Trace Definition Files (.TDF)

Static tracing occurs when a program developer has coded an API call to the system trace interface, which means you cannot specify at what point in the program flow tracing occurs, nor can you control what data is collected.

Trace Format Files are used by TRACEFMT. They specify how the trace data should be formatted. The filename implies which major code is described, and TRACEFMT generates the filename for the .TFF file from the major code of the event about to be formatted. If no description is found, or if the description does not describe all of the trace entry, TRACEFMT defaults to hexadecimal bytes for a default formatting. This will be covered in detail by hands-on exercises.

Trace Definition Files are used for dynamic tracing, and specifying one of them requires you to name the .DLL involved, or KERNEL. You may optionally a type or list of types and a group or list of groups. The .TDF file is used by TRACE to define where to collect data, and what data to collect.

Dynamic tracing occurs when trace definition files (.TDF) are used by the TRACE command. The implementation is that OS/2 inserts actual breakpoint instructions at the specified locations, and collects the data specified when the breakpoint is executed. There is no overhead for dynamic tracing when it is not in use, and a technician can be very creative when defining where to collect trace data, and what data to collect. We will create custom dynamic trace entries during hands-on exercises.

The OS/2 static tracepoints do not have associated TDF files, but may have associated TFF files that are used by the TRACEFMT.

Dynamic Trace Processing

Dynamic tracepoints are implemented as Trace Definition File (.TDF) entries. The TRACE command can be used to insert (and turn on) a dynamic tracepoint by patching it into its corresponding software module. Dynamic tracepoints are specified by the dynamic link library (DLL) filename and minor code.

Individual dynamic tracepoints can be qualified by separate type and group qualifiers. These qualifiers exist so that you can more easily turn on and off sets of related dynamic tracepoints. For example, all the dynamic tracepoints that are associated with pre-invocation events might have a type of PRE. Similarly, all the dynamic tracepoints that are involved in semaphore processing might have a group of SEM. In the TRACE command syntax, group is considered to have a stronger binding than type. This means that you can ask to turn on all events that are of a specified group that are also of one or more specified types. You do not need to use these qualifiers; they are there simply to make it easier to control related sets of dynamic tracepoints.

TDF files are typically found in the \OS2\SYSTEM\TRACE directory. They are identified by .TDF file name extensions. There are also Trace
Formatting Files (TFF) found within that directory. These files are used by the OS/2 Trace Formatter (TRACEFMT) utility to format the entries that are logged within the system trace buffer.

Commonly Used Abbreviations for OS/2 Groups and Types

<table>
<thead>
<tr>
<th>Groups</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>file system</td>
</tr>
<tr>
<td>LDR</td>
<td>resource loader</td>
</tr>
<tr>
<td>LNK</td>
<td>environment management</td>
</tr>
<tr>
<td>MOU</td>
<td>mouse I/O</td>
</tr>
<tr>
<td>MSG</td>
<td>message management</td>
</tr>
<tr>
<td>MSP</td>
<td>virtual memory management</td>
</tr>
<tr>
<td>NLS</td>
<td>national language support</td>
</tr>
<tr>
<td>PIP</td>
<td>pipe support</td>
</tr>
<tr>
<td>SEL</td>
<td>selector-related</td>
</tr>
<tr>
<td>SIG</td>
<td>signal handling</td>
</tr>
<tr>
<td>TIM</td>
<td>timer support</td>
</tr>
<tr>
<td>TSK</td>
<td>task management</td>
</tr>
<tr>
<td>VIO</td>
<td>video I/O</td>
</tr>
<tr>
<td>VM</td>
<td>virtual memory management</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>INT</td>
<td>internal</td>
</tr>
<tr>
<td>PRE</td>
<td>pre-processing invocation</td>
</tr>
<tr>
<td>POST</td>
<td>post-processing invocation</td>
</tr>
</tbody>
</table>

OS/2 Predefined Dynamic Trace Events

The file SYSTEM.TDF file supports dynamic tracing for the following:

**TRACE ON KERNEL**
- Major Code: 5 (decimal) 5 (hex)
- Groups: FS, LDR, NLS, PIP, SEL, SEM, SIG, TIM, TK, VM
- Types: PRE, POST, API, INT
- Purpose: Tracepoint definitions for APIs in the OS/2 kernel

**TRACE ON DOSCALL1**
- Major Code: 16 (decimal) 10 (hex)
- Groups: FS, LDR, LNK, MSG, MSP, NLS, SEM, TSK
- Types: PRE, POST, API
- Purpose: Tracepoint definitions for APIs in DOSCALL1.DLL

**TRACE ON MONCALLS**
- Major Code: 16 (decimal) 10 (hex)
- Groups: TSK
- Types: PRE, POST, API
- Purpose: Tracepoint definitions for APIs in MONCALLS.DLL

**TRACE ON QUECALLS**
- Major Code: 22 (decimal) 16 (hex)
- Groups: None
- Types: API, PRE, POST, INT
- Purpose: Tracepoint definitions for APIs in QUECALLS.DLL

**TRACE ON SESMGR**
- Major Code: 23 (decimal) 17 (hex)
- Groups: None
- Types: API, PRE, POST
- Purpose: Tracepoint definitions for APIs in SESMGR.DLL

**TRACE ON OS2CHAR**
- Major Code: 24 (decimal) 18 (hex)
- Groups: KBD, MOU, VIO
- Types: API, PRE, POST
- Purpose: Tracepoint definitions for APIs in OS2CHAR.DLL

**TRACE ON PMSHAPI**
- Major Code: 192 (decimal) C0 (hex)
- Groups: None
- Types: None
- Purpose: Tracepoint definitions for APIs in PMSHAPI.DLL

**TRACE ON PMWIN**
- Major Code: 194 (decimal) C2 (hex)
- Groups: None
- Types: None
- Purpose: Tracepoint definitions for APIs in PMWIN.DLL

**TRACE ON PMGRE**
- Major Code: 195 (decimal) C3 (hex)
Steps to Diagnose a Hang

Problems which are called 'hangs' fall into several categories.

The term 'hang' has come into use because there is frequently no way for a user of OS/2 to determine whether the problem is a loop or a wait. The term 'hang' is used in a generic way to mean 'the system does not respond as I expect', or 'I am unable to interact with the system'. The problem may be a loop, or it may be a wait.

Diagnosing any 'hang' will likely be much quicker if the system trace was used to collect appropriate data related to the symptoms.

Steps to Diagnose a Wait

Waits are recognized by the fact that no thread is ready. If the scope of the problem is a single application, we need only find out which thread is expected to run, and then analyse why it will not. If we cannot find out which thread we expect to run, we will need to do the analysis for each thread in the process, which will take somewhat more effort. Frequently, the application can be removed by using the window list to end it. If this has been attempted, and has not worked, one must find out why thread 1 will not execute.

If the scope of the problem is the user interface, one needs to examine it, as discussed above. The workplace shell is discussed elsewhere; remember that from the kernel's viewpoint, it is 'just another application'. This used to be a much more common symptom than in relatively current releases. It was typically noticed on a LAN server, when requesters received normal service responses, but the system administrator could not use the keyboard or mouse.

Frequently, if you haven't a well defined place to start, it works reasonably often to look at the blocking data for all threads, and to choose a resource which is needed by many threads. Pragmatically, if that resource could be made available, many threads would unblock. Therefore, choose one of the more 'popular' block ID's, and proceed to find out what thread owns it, why that thread will not run, and so on. You may need to do this for only one or two resources before you discover the key thread, and can focus your efforts on it.

If the scope of the problem is that OS/2 refuses to run any thread, the problem must be extremely basic, for example, the drive containing SWAPPER.DAT is no longer available due to a hardware problem, or the system has actually terminated, but has been unable to display that fact.

Steps to Diagnose a Loop

Loops are also relatively easy to recognize. When one inspects the collective status of all threads in the system, one thread will be in 'run' status, (if an SMP, one on each processor) and it is likely that many more threads are ready. If the priority of the thread is normal, an application may loop for a long time without the user being aware of the loop, although system performance may suffer somewhat.

To analyse a loop, follow one iteration of it. This is much easier to do with an interactive debugger than it is in a dump.

If the priority of the 'run'ning is in the time-critical class, the dispatcher is designed to prevent OS/2 from dispatching other threads. The looping thread is the cause of the problem, unless the loop is the correct response to another problem. In this case, contact the developer to find out why the thread must be such a high priority, and while you are talking, ask what could cause it to enter a non-ending loop. To diagnose a loop, use an interactive debugger to step through the loop, and try to understand what each conditional jump is really trying to accomplish. You can use an interactive debugger to lower the priority of an offending thread, and 'observe the results'. Recognise that this is quite legitimate, but that the application's integrity may actually depend on the behaviour you have just altered.
Serialization and Priorities in OS/2

This section describes the various ways to serialize access to resources, which is often required in a pre-emptive multitasking environment.

Brute Force Serialization

There are several serialization methods which attempt unilateral control over the dispatcher. Each has its own advantages and disadvantages. They will each be explained here before going on to semaphores, which are much more granular, and therefore less intrusive than these serialization methods.

Uniprocessor Method - Disable Interrupts

There is a way for privileged code to guarantee serialization in a single-processor environment, namely to disable interrupts during the actual inspection and update of the protected resource, and then to enable interrupts promptly. The overhead of this method is practically nil, but it is potentially dangerous because it disables pre-emption, which reduces the responsiveness of the system to the user. It also represents a barrier to running successfully on a multiprocessor, because all other processors are unaffected by this, and it therefore requires the developer to re-examine parts of a program which are no longer serialized, but may well be thought to be properly serialized.

Multiprocessor Methods - Spin Locks

In a multiprocessor environment, there are additional problems, namely how to control the additional processors, which may be executing exactly the same instruction at the same cycle. The solution to successfully serializing access to critical structures is solved by using a special instruction prefix, LOCK, which guarantees that all accesses to memory for the following instruction occur as a unit, with no intervening cycles by other processors, DMA devices, or bus masters.

Instructions such as:

- Increment(INC), Decrement(DEC), Add(ADD,ADC), Subtract(SUB,SBB),
- Logical operations, (AND,OR,XOR,NOT,NEG),
- Exchange(XCHG), Exchange&Add(XADD), Compare&Exchange(CMPXCHG)

can be used to claim a resource, add a node to a linked list, and perform other atomic events which normally require serialization, like selecting a ready thread to run.

Note: The 486 and following processors assert the hardware signal 'lock' for XCHG, XADD and CMPXCHG instructions if an operand is in memory, regardless of a prefix.

Each processor will use the appropriate method to attempt its task, and if the condition code does not indicate success, it will simply retry the operation until it does complete. This is called a 'spin loop', and this method of serialization is called a 'spin lock'. It is used when it is expected to be able to access the lock in less time than it will take to save the current context and find another thread to run.

One should not expect to discover the presence of spin locks in a non-multiprocessor environment, because they should always be available, and the spinning should never occur.

DosEnterCriticalSection & DosExitCriticalSection
These API's will serialize all threads in a process. They have no effect on threads in other processes. At the time control returns from DosEnterCriticalSection, no other thread in the process is allowed to execute (there is an exception to this which is when a signal is sent to a process). Looking at the threads' status, one may see 'crt'. This is NOT the thread that entered critical section. Threads in the 'crt' state are ready to run but are temporarily held because of the existence of the thread in critical section. The critical section thread may block, but is the only thread in that process which is allowed to run. Only when that thread issues the DosExitCriticalSection are the other threads released, and again allowed to compete for use of the processor. This is really too much serialization for most situations, because it temporarily disables multithreading in the process, regardless of the other threads' design, or current actual processing.

DosSuspendThread & DosResumeThread

Some applications are designed such that there is a limited number of threads which will access some shared resource, and others never will.

To access a resource in a protected way, one can simply suspend the other threads which represent a possibility of simultaneous update, and leave the remaining threads alone. This is therefore less intrusive than the critical section API's, but still may affect threads which do not represent a threat at this instant, due to other processing, timing, and so on.

DosSuspendThread API will cause a specific thread to no longer compete for the processor, until DosResumeThread is issued. A thread in this situation will have the status of 'frz'. It may not be possible, without an appropriate trace, to find out which thread suspended another.

Semaphores

The least intrusive way to guarantee serial access to a shared resource is to associate a semaphore with it, and to acquire ownership of the semaphore before accessing the resource. The application threads will be suspended only when there is actual contention for the resource.

This does require all of the programmers involved to be careful to request the semaphore before accessing the resource, and to remember to free it when done. The classic solution to this is to build a low-level function which includes the serialization.

Semaphores are of three categories:

1. Kernel Semaphores, or KSEMs.

KSEMS will be discussed later, because we will focus first on items available to the application programmer.

2. 16-bit semaphores.

There are two basic kinds of 16-bit semaphores, and an add-on structure which makes a third type by aggregation.

They are the System Semaphore, the RamSem, and the FastSafe RamSem, which is an accounting structure prefixed onto a RamSem.

3. 32-bit semaphores.

There are two types of 32-bit semaphores, Mutual Exclusion, or MutEx and Event Semaphores. It is also possible to wait on a list of EventSems or MutexSems, but all semaphores in a list must be of the same type.

16-bit Semaphores

There are three types of 16-bit semaphores, namely system, RAM, and fast-safe RAM semaphores. There are compromises involved in using each.

System Semaphores

These are the most robust of the three, and have the most overhead.
One thread must create the semaphore, with `DosCreateSem`, which has a name in a format similar to a file name, but in root directory `SEM`. Other threads must open it with `DosOpenSem` to get its handle.

Use is to issue `DosSemRequest`, use the resource, and then to issue `DosSemClear` so that other threads can access the resource. All threads should issue `DosCloseSem` before ending.

If a thread ends while owning a system semaphore, the first requestor is given a return code that indicates the situation, so that it is warned of a possibly incomplete update, and may take whatever action is necessary to recover, or terminate.

To find out which thread owns a system semaphore, display a word at the address provided in the blocking data. The address will be a logical address using a GDT selector, generally 400:xxxx. The 12 low order bits are the slot number of the thread which owns the semaphore. If unowned, the value is zero.

**RAM Semaphores (RamSems)**

At the opposite end of the scale is the extremely fast RamSem. Most of the speed comes from the following facts:

- API's use the address of the RamSem as the handle.
- OS/2 assumes a RamSem is local to a process.
- OS/2 does absolutely no accounting for a RamSem.
- OS/2 can not provide any recovery for a RamSem.

A RamSem is owned by a user thread if the first byte is hex 'FF', otherwise it is not owned by a user thread. Unless you have a trace, there is no way to determine which thread owns a RamSem.

**Fast-Safe RAM Semaphores (FSRamSems)**

The FSRamSem is a compromise between the two earlier types.

The FSRamSem is nothing more than a structure which includes a RamSem. The fields of the structure record the process ID (PID) and thread ID (TID) of the thread which owns the semaphore, or zero if unowned. They also include a use count, which is incremented if the owning thread again requests the semaphore. This allows recursive functions to serialize without being blocked, waiting for a resource the thread already owns.

The `DosFSRamSemRequest` API is used to request the semaphore. It returns when the resource is owned by the thread.

The `DosFSRamSemClear` API is used to release the semaphore. If the use count is not zero after being decremented, the semaphore is NOT released. There must be as many 'Clear' as 'Request' API calls to actually release the semaphore, and allow other threads to compete for it.

**32-bit Semaphores**

There are two classes of 32-bit semaphores, private and shared. There are three types of semaphore in each class, Event, MUTual EXclusion, and multiple wait semaphores.

**MUTEX** semaphores correspond to one of the most common uses of the 16-bit semaphores, namely to allow competing threads to mutually exclude others from accessing a shared resource.

A MUTEX semaphore includes the slot number of its owner, if owned, or zero if unowned.

An EVENT semaphore contains a 'post count' which is incremented each time it is POSTED, and decremented each time a WAIT for it is completed successfully. This type provides a way to assure that some processing occurs exactly once for each POST.

A multiple wait semaphore is nothing more than a list of semaphores, of the same type. A thread may wait on either 'ANY' or 'ALL' of the semaphores in the list.

All Semaphores must first be created with `DosCreate???Sem`, where '???' is the semaphore type. Other processes must open them with `DosOpen???Sem` to have access to them. Private semaphores have a null pointer to their name, and thus no name. Public semaphores have a name in the same format as that used for the 16-bit semaphores. `DosClose???Sem` is used when a thread is through using it.
DosRequestMutexSem and DosReleaseMutexSem are used to access the mutual exclusion semaphores.

DosPostEventSem and DosWaitEventSem are used to access an event semaphore. DosResetEventSem will allow immediate access, and return the post count, which is cleared by this API.

DosQuery???Sem will allow the retrieval of information about each type of semaphore.

DosAddMuxWaitSem and DosDeleteMuxWaitSem are used to add and delete semaphores from a multiple wait semaphore list, respectively.

Dispatching Priorities

This section describes how the priority of a thread is set, and defines what the classes mean for debugging.

The Dispatcher, Priorities, and Dispatching Classes

The dispatcher's task is to give control to the proper thread. The definition of 'proper' thread can be difficult to state. My approach to this problem is to state the obvious cases, and then to focus on what is left. In a sense, this discussion parallels the logic in the dispatcher.

1. Idle Class.

No other class will be pre-empted in order to run an idle class thread. The notion of starved, and the MAXWAIT parameter do NOT apply to Idle Class threads. OS/2 by design will not execute a ready Idle Class thread as long as threads in other classes are ready.

2. Regular Class.

Most threads are expected to be in this class. All dispatching options and parameters apply to scheduling this thread.

3. Time-Critical Class.

As long as any thread in this class is ready, OS/2 will give control to it. By design, this may prevent threads in other classes from running. You cannot use priority as a serialization method.

For example, a page fault will result in temporarily blocking this priority thread.

4. Fixed-High, or Server Class.

The threads in this class are at a somewhat higher priority than those in the regular class which do not have the focus, but below time-critical.

Note: The numbers above are what the programmer specifies in DosSetPriority, or the 16-bit API DosSetPrty, and are what is returned by DosGetPriority. OS/2 processes these class numbers to create an internal dispatching priority. There are 32 priority levels in each class, which range from 00 to 1F. The priority levels, or deltas, stay the same as the programmer specified initially, if PRIORITY=ABSOLUTE is specified.

The internal priorities have a range from 01 to 08, with 01 usually used for idle-class threads, and 08 usually used for time-critical threads. If PRIORITY=DYNAMIC was specified or defaulted, there are priority boosts given for the following reasons:
Being the foreground process; and for owning the keyboard;
Yielding the processor before the end of the time slice
IO completion
Being 'starved', that is, ready status and not dispatched for MAXWAIT seconds.

Dispatching is the process of finding the correct ready thread, and then giving control to it. Within each class, the priority delta is used to choose which thread should have control. When several ready threads have the same priority, control is given in turn to each of them, based on the TIMESLICE parameter. The minimum value of this parameter is the duration of the priority boosts which may be applied. The maximum value is the longest a thread can execute before being pre-empted for other threads which have the same internal dispatching priority.

As long as a group of threads at some priority use all the processor, control is not given to lower priority threads. What happens is that the other waiting threads become 'starved' after MAXWAIT seconds have elapsed, and their priority increases until they receive at least a minimum timeslice.
Idle-class threads are never starved, and so will not receive this boost.

Note:
When running in the kernel and device drivers, pre-emption can not occur. Threads must explicitly give up their time-slice to give other threads an opportunity to run.

How to Display Dispatching Priority

Use the '.p' command on the slot of interest, and find the pTCB, which is the linear address of the Thread Control Block.

For slot F, below, we see the following FOR .p output:

```
Slot Pid  Ppid CsId Ord Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
000c 0005 0000 0005 0001 blk 0200 7cf7f000 7d1858a4 7d16a0d8 1eb8 00 pgma
0008 0005 0000 0005 0002 blk 081f 7cf77000 7d1858a4 7d169a28 1ea8 00 pgma
000e 0005 0000 0005 0003 blk 021f 7cf83000 7d1858a4 7d16a430 1ea8 00 pgma
000f 0005 0000 0005 0004 blk 061f 7cf85000 7d1858a4 7d16a5dc 1ea8 00 pgma
0010 0005 0000 0005 0005 blk 0200 7cf87000 7d1858a4 7d16a788 00 pgma
000d 0006 0000 0006 0001 blk 0200 7cf81000 7d1860d0 7d16a284 1eb8 00 pgmb
000a 0006 0000 0006 0002 blk 021f 7cf7b000 7d1860d0 7d169d80 1eb8 00 pgmb
0013 0006 0000 0006 0003 blk 0800 7cf8d000 7d1860d0 7d16ac8c 1eb8 00 pgmb
```

CAUTION: the offset used is correct for WARP CONNECT, but the addresses are what were used in OS/2 2.11, so this is a somewhat mixed example. Any offset in any control block may change any time a fix or new version is installed. Please refer to the Thread Control Block in the System Diagnostic Reference for offsets relating to other versions.

The first byte is the programmer's priority class, ranging from 1 to 4. The second byte is the level within the class, ranging from 00 to 1F. The third and fourth bytes are not useful. The fifth and sixth bytes are OS/2's computed dispatching priority. This field is a word, so the high order part is byte 6.

081F is the highest possible value.
0100 is the lowest possible value.

On a uniprocessor, using DosSetPriority to make yourself time-critical at the highest delta value would give you an extremely good chance of not being pre-empted, and was occasionally misused for serialization. This will never work on a multiprocessor, and is risky even on a uniprocessor, because a page fault will cause you to lose control while the page is processed, just as doing I/O to a file will cause a thread to block if access to the actual device is required.

The Status of a Thread

A thread can be in one of several states. The following list is an attempt to list all the possible states, and to briefly discuss each.

run This thread is currently executing.
rdy This thread would like to run, but higher priority thread(s) are executing, which prevents this thread from running.
blk This thread is blocked. Use the '.pb' command to find out what resource (block id) it is blocked on (waiting for).
crt This thread cannot be run because another thread in this process is currently in a critical section.
frz This thread is frozen, that is, some other thread has called the API DosSuspendThread and passed the ID of this
thread. This thread cannot execute until some thread issues DosResumeThread to inform OS/2 that this thread is once again eligible to be dispatched. There is no way to discover what thread suspended it.

A Form to Use For Unwinding Stacks

<table>
<thead>
<tr>
<th>Frame at</th>
<th>Caller's Frame</th>
<th>Return Offset</th>
<th>Selector</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
<tr>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
<td>______</td>
</tr>
</tbody>
</table>

Advanced Guide to Hang Analysis

What is a hang?
It's an external symptom or a user perception that little or no work is being done. It could be a case of extremely poor performance. The hang symptom categorises itself into three distinct cases:

**Wait**

Threads and processes are not being dispatched by the Operating System. Thread status gives the initial clue. Use of the `.P` command determines status.

**Blocking**

Threads may wait voluntarily for a resource or an event, in which case they will probably be Blocked. They might poll for the resource, in which cases they will cycle through blocking, being ready and running until the resource becomes available.

A notoriously problematic case of Blocking is the deadlock. This is where two threads are each own exclusively a resource and block waiting for ownership of the other's resource.

**Suspension**

Another thread may have deliberately debarred a thread from being scheduled, in which case we will see the waiting thread in a `cr` or `frz` state.

**Pre-emption**

Another thread monopolises the system. Typically the hanging threads will be ready for dispatch (`rdy`), but will never or rarely receive a minimum time-slice. We look for running and ready threads with an excessively high priority. The `.P` will give the calculated priority of each thread.

**Disabled wait**

Looks rather like a H/W Freeze. Last instruction executed was a HLT having sometime previously disabled interrupts using CLI. This usually happens only when ring 0 code detects a terminal condition. One would hope that some form of diagnostic information had been displayed prior to this particularly if NMI have been disabled also! If NMI have not been disabled then an artificially generated channel check may be used to cause an NMI, which would allow one to break into the kernel debugger. However the KDB to allows only one NMI channel check per boot. If NMI are disabled then H/W analysis techniques may be the only recourse.

**Loop**

A thread is running more or less permanently. It's state will mostly be `rdy` or `run`. Similar analysis techniques to trap apply here. We examine the registers of the running thread by using `.R`. From the we can determine in which module the thread is running by looking at the owner of the executing code segment. If necessary we unwind the stack and determine the caller etc.. Analysis is very similar to trap analysis.

**H/W Freeze.**

The processor fetch-execute cycle has been suspended. Not even an NMI interrupt will resume instruction fetch. This is almost certainly a hardware problem. Timing/clocking problems caused by incompatible cards, overloaded busses, incorrect bus terminations, faulty processor or controller chips. Use H/W techniques such an ICE machine or Logic Analyser.

The following Theory Topics are now covered in detail:

- **The Wait Condition - Diagnostic Techniques**
  This is further subdivided into two topics of discussion:
  - Memory Management and Ownership Topics
  - Thread Scheduling and Dispatching
- **Program Design Issues and Weaknesses**

The final section of this Guide is a collection of Worked Examples that explore memory management, the File System, Presentation Manager and Ring 0 Loops From a Dump.
In most problem analyses the question of memory ownership or use will arise. For example:

- To which module does this instruction belong?
- Which process executed this module?
- Who allocated this storage?
- Who passed these parameters?
- What control block does this address point to?

In fact the frequency with which this question is asked makes it a fundamental aspect of analytical technique.

For hang analysis this is no less true:

- In the case of loops, analysis proceeds in a similar manner to that of traps.
- In the case of waits, a key piece of information is the BlockId. In many cases this is an address of a system control block that relates closely to the reason for waiting. Discovery of the owner of storage pointed to by a BlockId is therefore of prime interest.

We start by reviewing memory management in OS/2 and in particular memory ownership.

Memory Management and Ownership Topics

Memory allocation in a demand-paging virtual storage operating system such as OS/2 embodies the allocation of a number of system resources with certain attributes applied:

**Resources**

- Virtual address space
- Real address space
- Auxiliary address space (SWAPPER)

**Attributes**

- Exclusion (privacy)
- Inclusion (sharing)
- Owner (Where is was allocated from or who it was allocated to)
- Requestor (who made the request on behalf of the owner)
- Permissions (Read-only, Read/Write, Executable)

Use of the resources and the attributes applied is tracked by the system in its VM control blocks. The most important of these are:

- **VMAH**: The Virtual Memory Arena Header Record
- **VMOB**: The Virtual Memory Object Record
- **VMAR**: The Virtual Memory Arena Record
- **VMCO**: The Virtual Memory Context Record
- **PF**: The Page Frame Structure
- **VP**: The Virtual Page Structure

Virtual Address Space Arenas and Regions
OS/2 partitions the 4G virtual address space into three types of arena:

- System
- Shared
- Private

The system arena is common to all processes. It starts at the 512M boundary and occupies the address space up to 4G. Only system code (and device drivers) can access data in the system arena directly. User code must use APIs invoked by the call gate mechanism to access system arena code and data. Nearly all system arena data is global: that is, managed by a common set of page tables, whatever the current thread/process context. The exception to this is in the memory area mapped by selector 30. Page table entries are adjusted as part of context switching so that selector 30 addresses the current PTDA, TCB and TSD.

The shared arena address range is common to all processes, but it comprises data that is both global and instance. Instance data occurs where a separate set of page table entries are used per context to map the same linear address range.

Instance data is used when the same type of data needs to be allocated as multiple private copies to each process. An example of this would be a logical screen buffer. The shared arena starts initially at the 304M boundary and ends at 512M. User programs may access the shared arena. DLL code and data is located in the shared arena. DLL code segments are always global, but DLL data segments may be instance or global and are usually a mixture of both.

The shared arena is further subdivided into a number of regions:

<table>
<thead>
<tr>
<th>Region</th>
<th>Description</th>
</tr>
</thead>
</table>
| Protected | This region is reserved for protected data segments of protected DLLs. In General 16- and 32-bit applications do not have addressability above the 448Mb boundary. Potentially 32-bit applications are able to modify all read/write global data, whether intended by the owning DLL or not. The Protected region is provided so that Protected DLLs can isolate their data from general access. Only Protected DLLs have addressability to the protected region by being assigned data selector 63. 32-bit DLLs become protected through use of the protect option at compile time. 16-bit DLLs may also use the protected region, if explicitly coded to do so and listed in CONFIG.SYS using:  

```
PROTECT16=dll1,dll2,....
```

The Protected Region may be subsumed into the Based Region (see below) by coding in CONFIG.SYS the NOPROTECT option on the MEMMAN statement.

The default is MEMMAN=PROTECT

**Note:**

From OS/2 Warp V3.0 fix pack 19 and OS/2 Warp V4.0, the Protected Region has been absorbed into the Based Region. The system behaves as if MEMMAN=NOPROTECT is in effect and MEMMAN=PROTECT has no effect.

| Based | The Based Region is reserved for non-protected DLLs that have a preferential base address assigned by the linkage editor by using the BASE option. The purpose of the Based Region is to improve performance of module loading, by avoiding the need for the System Loader to do fix-up processing. |

**Note:**

Under OS/2 2.x, MEMMAN=NOPROTECT would cause the Based and Packed Regions to move up 64M bytes - effectively giving another 64M bytes for general purpose use in the Shared Arena.

| Packed | The Packed Region is reserved for 16-bit DLL code segments. Within the Packed Region the Compatibility Region Mapping Algorithm does not apply. Code segments are packed contiguously to make best use of physical pages. Potentially, tiny DLL code segments can deplete physical storage very rapidly if not packed. However, when packing is used there is no general algorithm that will convert 16-bit addresses into 32-bit addresses within the Packed Region. The system has to scan |
the LDT, over the Packed Region, to make this conversion.

Packing may be disabled by specifying the **NOPACK** option of the **MEMMAN** statement in CONFIG.SYS. The default is **PACK**. When packing is disabled up to 32M bytes is made available to the Global Shared Region.

**Notes:**

Under OS/2 2.x only **MEMMAN=NOPACK** would tend to reduce the Swapper Size where a great many 16-bit code segments are in use. This is because code segments outside the Packed Region are normally discardable (they are not swapped). Within the packed region they are swappable since a 4K page may contain code from a number of different modules.

Under OS/2 2.x **MEMMAN=NOPACK** would provide up to 32M bytes extra virtual address space for general purpose use in the Shared Arena.

Packing does not affect the availability of LDT selectors for allocations in the Packed Region, just the base linear address boundaries on which they are deployed.

Packing should not be confused with either the LINK386 PACK option or the PACK.EXE utility.

From OS/2 Warp V3.0 fix pack 19 and OS/2 Warp V4.0 the packed region has been reduced to 16Mb.

**Global Shared**

This region has a lower boundary at 320M bytes and includes the Packed, Based and Protected Regions. This is reserved for Global Read-Only allocations only. Since no Read/Write data is allocated in the Global Shared Region some page table economies are possible. Also process context switching performance is improved.

**Notes:**

The Global Shared Region is not configurable.

The Global Shared Region is only implemented in OS/2 WARP version 3.

Under OS/2 2.x Read/Write segments would be allocated in the Based Region.

**ReadWrite Basing**

The Read/Write Basing region is the preferred region for locating Read/Write DLL data segments where a base address has been assigned to a DLL module by the linkage editor. The purpose of this region is to keep Read/Write segments out of the Global Shared Region and thus retain its performance advantages. It also places an upper bound on the location of dynamic shared allocations, namely the Minimum Read/Write Basing Region address.

**Notes:**

The Read/Write Basing Region is not defined in OS/2 versions prior to version 3.

Based DLLs under OS/2 2.x, by preference, have their segments loaded sequentially starting with segment 1 at the base address. With the implementation of the Global Shared Region only Read-Only segments can be loaded sequentially from the base address.

**Expansion**

The Shared Arena is an expand-down arena, that is, allocation searches for free regions from the high addresses to low. The Shared Arena therefore expands from the minimum Read/Write Basing Region address towards the highest upper bound of all the Private Arenas. This area is the expansion region for both the Shared Arena and all the Private Arenas.

The Shared Arena will not contract to an address higher than the minimum Read/Write Basing address.

**Note:**

The expansion region for OS/2 2.x is from the lower boundary of the Packed Region, if present. If not, then from the lower boundary of the Protected Region, if present. If both the Protected and Packed Regions are removed (using **MEMMAN=NOPACK,NOPROTECT**) then expansion occurs from the top of the Shared Arena.
Each private arena occupies the lowest range of virtual address space from 0 - 64M bytes expanding up to a maximum of 304M bytes, the minimum Read/Write Basing address. None of the Private Arenas will be allowed to expand beyond the lowest allocation in the Shared Arena, that is Private and Shared Arenas may not overlap.

In general each process uses a separate set of page table entries to map each page of its private arena. Thus the data in the private arena is private to each process. Code (.EXE files) however is treated differently. Since code is read only an economy is made whenever more than one process runs the same .EXE. Where this happens the same page table entries are used among the processes sharing the common .EXE file. User programs may only access the private arena of the process they are running in (a special exception to this is possible through the DosDebug API by defining memory aliases).

Virtual Memory Arenas and Regions may be presented pictorially as in the following diagram.

**Note:**

Some regions of the 4G address space are reserved. This is done for a variety of reasons which include:

- H/W and BIOS restrictions
- Enforced segregation between Arenas
- Guaranteed reserved address ranges.
Virtual Address Space Regions
(OS/2 4.0 and 3.0 Fix Pack 19)

System Arena

Reserved Regions

Shared Arena
- non-based code

Based Region

Packed Region

non-based code

non-based data + overflow

ROW Basing Region

Expansion Region

Private Arena process 1

Private Arena process 2

Private Arena process 3 (VDM)
Virtual Address Space Management

Each of the three types of arena discussed in the previous section is managed by:

- An Arena Header Record (VMAH)
- A Sentinel Arena Record (VMAR)

The VMAHs are maintained in a double-linked list. They contain information about the extent to which an arena has been used. Of particular interest are the following fields:

+0x0 Pointer to the next VMAH
+0x4 Pointer to the previous VMAH
+0x8 Pointer to the Sentinel Arena Record for this arena
+0xc Pointer to the VMAR adjacent to the 1st free area.

- In the case of expand down arenas (the shared arena), this is the VMAR for the region of memory allocated above the first free area below the Minimum Read/Write Basing region.
- In the case of expand-up arenas (system and private) this is the VMAR for the region of memory allocated just below the lowest free area.

+0x20 Current minimum linear address allocated.
+0x24 Current Maximum linear address allocated.

VMAHs are located:

- at _ahvmSys for the Shared Arena
- at _ahvmShr for the High Memory System Arena
- at _ahvmShr for the System Arena
- imbedded at +0x40 in each PTDA for Private Arenas

Arena Records (VMARs) are used to describe virtual storage reservations. These are described in more detail in Virtual Memory Arena Records, below.

A special form of the VMAR is the Sentinel Arena Record. This serves two purposes:

- To track the theoretical size limits of an Arena
- To act as the head to a double-linked list of Regular VMARs, each of which describes a specific allocation.

The sentinel VMAR for the Shared Arena is called the Boundary Sentinel, since it determines where the (dynamic) boundary between shared and private arenas lies. The boundary is adjusted to reflect the current highest private arena address.

The manner in which VMARs and VMAHs are organised to manage the three types of arena is shown in the following diagram:
Virtual Memory Arena Records

VMARs are 24-byte records that describe virtual address space allocation, or reservation. They are located in a table in system memory. The principle use of the VMAR is to track the allocation of virtual memory, which may or may not be backed in RAM or on the SWAP file.

Arena records appear in a number of guises depending on the area storage they describe and whether the storage is shared, instance or private data. They are formatted by the KDB and DF .MA command. .MA takes either the handle or address or the VMAR as a parameter, or if no parameters are specified then the entire VMAR table is formatted.

```
# .ma
har     par      cpg        va    flg next prev link hash hob   hal
0001 %feeef020 00000100 %ff050000 001 001f 0002 0000 0000 0001 0000 =0000
0002 %feeef036 00000161 %feeef000 001 0001 0003 0000 0003 0002 0000 =0000
0003 %feeef04c 00000100 %feeef000 001 0002 0021 0000 0003 0000 =0000
0004 %feeef062 00000000 %60000000 003 05d3 0015 0000 0000 ffc0 0000 max=%fffc0000
0005 %feeef078 0000cc40 %d4000000 007 0617 0072 0000 0000 ffe0 0000 max=%eff0000
0006 %feeef08e 00000003 %ffe1b000 009 000f 03ad 0000 0000 0007 0000 =0000
0007 %feeef0a4 00000000 %ffe22000 009 0008 01df 0000 0000 0008 0000 =0000
0008 %feeef0ba 00000000 %fee2e000 009 0008 0019 0000 0000 0009 0000 =0000
0009 %feeef0d0 00000010 %fee3b000 009 01e2 0008 0000 0000 000a 0000 =0000
000a %feeef0e6 0000001c %ffe124000 121 00e1 0008 0000 0000 0010 0000 =0000
000b %feeef0fc 00000000 %ffe24d000 121 00e1 0008 0000 0000 0010 0000 =0000
000c %feeef112 0000003b %ffe33000 009 0034 000b 0000 0000 000d 0000 =0000
000d %feeef13a 00000010 %11450000 379 0394 02d6 0000 0000 000e 0000 =0000
000e %feeef13e 00000001 %ffe10000 001 0083 0200 0000 0000 000f 0000 =0000
00b6 %feeefaf6 00000080 %00110000 169 01df 0076 0000 0000 041f 0000 hptda=03c9
```

The fields of principle interest are:

- **har**
  The arena record handle. This is a unique identifier assigned to each VMAR.

- **cpg**
  The number of pages (4K bytes) allocated or reserved.

- **va**
  The address of the first page in the reservation.

- **hob**
  The handle of the VMOB that occupies the virtual address range covered by va and cpg.

The right-hand column gives descriptive information about the use of the address range in a VMAR. Of particular interest are:

- **sel=ssss**
  Indicates system storage mapped by a GDT selector.

- **hco=hhhh**
  Indicated shared global storage. The hco is the handle of the VMCO at the head of the list representing accessors to an allocation in the Shared Arena.

- **hptda=pppp**
  Indicated private memory allocated in the private arena of a process whose PTDA handle is pppp.

Virtual Memory Object Records

VMOBs are 16-byte records allocated contiguously in a table in system memory. Each table entry is numbered from 1 and is referred to as a memory object handle, or more simple as a hob.
VMOBs are used to store information about the allocation request. Of particular interest are:

- The Requestor
- The Owner
- The Permissions

The VMOB also has links to other related control blocks. Of these the important ones are:

- The associated VMAR,
- the associated VMCOs,
- and associated VMOBs.

VMOB is formatted by using the KDB or DF .MO command. .MO takes either the handle or address of the VMOB as a parameter, or if no parameters are specified then the entire VMOB table is formatted.

```
##.mo
hob har hobnxt flgs own hmte sown,cnt lt st xf
0001 0001 fec8 0000 0ff1 0000 0000 00 00 00 00 vmob
0002 0002 fec8 0000 ffe3 0000 0000 00 00 00 00 vmar
0003 0003 fec8 0000 ffec 0000 0000 00 00 00 00 vmah
0004 %fff13238 8000 ffel 0000 0000 00 00 00 00 doscalls.dll
0005 %fff13190 8000 ffel 0000 0000 00 00 00 00 os2krnl
0006 %fff0a891 8000 ffa6 0000 0000 00 00 00 00 mte       doscalls.dll
0007 0006 0000 0000 fff6d 0000 0000 00 00 00 00 doshlp
0008 0007 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
0009 0008 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
000a 0009 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
000b 000a 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
000c 000b 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
000d 000c 0000 0325 ffb0 0000 0000 00 00 00 00 lock
000e 000d 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
000f 000e 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
0010 008f 0000 402c 00ae 0115 0000 00 00 00 00 priv 0002 c:pmshell.exe
0011 0010 0000 0000 ff37 0000 0000 00 00 00 00 romdata
0012 0011 0000 0000 ff8a 0006 0000 00 00 00 00 os2krnl
```

One VMOB is formatted per line with the hob in the left-hand column.

The owner is shown under the own column and is given as a hob that is associated with and uniquely identifies where the allocation is made from. For example:

Memory dynamically allocated within a Private arena uses the handle of the PTDA (hptda) as the owner.

The PTDA has a number of characteristics that make it an ideal choice for an owner:

- Each process has a unique PTDA
- The PTDA is the central control block from which all information about a process is obtained.
- Each PTDA is allocated from a unique memory object so has a unique hob, which defined to be the hptda.

For storage allocated for a load module segment the module MTE handle (hmte) is used.

The MTE has a number of characteristics that make it an ideal choice for an owner:

- Each loaded module is represented in the system by a unique MTE.
- Each MTE is allocated from a unique memory object so has a unique associated hob, which is defined to be the hmte.

Memory allocated in the shared arena which is not specific to a particular process uses the following conventions for owner:

- For DLL instance and global data the owner is the hmte of the owning DLL.
- For Giveable shared storage, the owner is (0xff5).
- For Gettable shared storage, the owner is (0xff6).
- For Giveable and Gettable shared storage, the owner is (0xff7).
For Named Shared Storage, the owner is (0xff82).

See DosAllocSeg, DosGiveSeg, DosGetSeg and DosAllocSharedMem APIs in the Control Program Programming References for OS/2 1.x, 2.x and 3.x.

Memory allocated or suballocated from the system arena uses an artificial system owner id (ffxx) that doesn't actually correspond to a VMOB but is a conventional handle used to indicate the type of system object which has been allocated. An example this is hob 1 which is the table of VMOBs.

The requestor's id is shown in the hmte column. This field is either:

- The hmte of the module making the request.
- An associated system object
- zero where there is no associated requestor.

To the right of each line appears a textual interpretation of the own and hmte fields.

The handle of the associated VMAR is shown in the har column.

Associated VMOB's that share the same virtual address (that is, instance data) are linked from the hobnxt field.

Not every VMOB is linked to an associated VMAR, as seen in hobs 4 and 5 in the example. These are known as pseudo-objects. They are used for some small system control blocks that are allocated, as required, from system storage but are too small to warrant the overhead of the minimum allocation of 1 page, which an arena records implies. PTDAs and MTEs are the most frequently encountered pseudo-objects.

The va field replaces the har and hobnxt and points directly at the object itself.

---

### The Virtual Memory Context Record

VMCOs are small control blocks that serve as extensions to the VMAR for shared arena, shared data. Whenever a process is given access to a shared global data object (most frequently DLL code and global data) then a VMCO is used to record the handle of the process (hptda) of the accessing process. Each process that shares a global data object will have a VMCO chained in a single-linked list from the object's VMAR.

VMCOs may be formatted using the KDB and DF .MC command, however they are usually displayed with their corresponding VMOB and VMAR by using either the .MOC or .MAC commands.

```
# .mac 297

*har par cpg va flg next prev link hash hob hal
0297 %feef2904 00000660 %11fb0000 369 0312 0295 0000 009e 02a1 0000  hco=0057f
  hob hobnxt flgs own hmte sown,cnt lt st xf
hob 02a1 0297 0000 4a2d fff5 0302 0000 00 00 00 00 give
  hco=057f pco=ffe70b96 hconext=00241 hptda=06d1 f=1e  pid=0059
hco=02a1 pco=ffe6fb60 hconext=004ce hptda=04b2 f=1e  pid=0043 c:pmsspool.exe
hco=004ce pco=ffe70821 hconext=0034c hptda=05c3 f=1e  pid=0016 c:pmawn.exe
hco=0034c pco=ffe70907 hconext=0014e hptda=04ca f=1e  pid=000f c:pmshell.exe
hco=004ca pco=ffe7080d hconext=00348 hptda=05c3 f=16  pid=0016 c:pmawn.exe
hco=00348 pco=ffe70083 hconext=0017a hptda=04ca f=16  pid=000f c:pmshell.exe
hco=0017a pco=ffe6f77d hconext=00177 hptda=039f f=16  pid=000b c:psdaemon.exe
hco=00177 pco=ffe6f76e hconext=00148 hptda=03ec f=16  pid=000c
hco=00148 pco=ffe6f683 hconext=000b2 hptda=03c9 f=16  pid=000a c:dddaemon.exe
hco=000b2 pco=ffe6f395 hconext=00083 hptda=0359 f=16  pid=0009 c:harderr.exe
hco=00083 pco=ffe6f2aa hconext=00081 hptda=02e1 f=16  pid=0007 c:landll.exe
hco=00081 pco=ffe6f2a0 hconext=0007d hptda=02ad f=16  pid=0006 c:laasmsg.exe
hco=0007d pco=ffe6f28c hconext=00337 hptda=027a f=16  pid=0008 c:pmshell.exe
hco=00037 pco=ffe6f12e hconext=00000 hptda=02ac f=16  pid=0004 c:cgambit.exe
```

---

### Private Arena Private Data

Private data, that is data in a Private Arena not accessible from any other context, is managed by VMARs and VMOBs as depicted by the
following diagram.

Control blocks and data that directly represent the allocation are shown shaded.
Private Arena Shared Data

This is the case where .EXE program Read/Only segments are shared across multiple Private Arenas.

The following diagram depicts this situation.

Note that only one VMOB is used, but there are multiple VMARs, one for each process accessing the allocation. Each VMAR is linked using the link field.

Control blocks and data that directly represent the allocation are shown shaded.
Shared Arena Global Data

DLL Global Data and Named Shared, Giveable and Gettable allocations are potentially shareable among multiple processes. With these types of allocations data and address range is common to all who access it. Those that are given access are recorded by the VMCO chain.

With this type of allocation, there is only one VMAR, and VMOB.

Note that the own field of the VMOC, which is interpreted on the right hand side of the .MO display, may be one of five possibilities:

- **hmte** Data is Global Data or Code segment of a DLL
- **Give** Data is allocated with the Give attribute
- **Get** Data is allocated with the Get attribute
- **GiveGet** Data is allocated with both the Give and Get attributes.
- **Mshare** Data is named shared storage.

The following diagram depicts this situation.

Control blocks and data that directly represent the allocation are shown shaded.
Shared Arena Instance Data

A DLL instance data allocation shares only its address range among its accessors. The data is mapped to a different set of physical pages for each process.

This type of allocation is represented by a single VMAR with a chained list of VMOBs, one for each accessor. This is the only case where VMOBs are linked by the `hobnxt` field.

The following diagram depicts this situation.

Control blocks and data that directly represent the allocation are shown shaded.
Shared Arena Instance Data

Shared Arena

Instance data (virtual)

VMAR

physical mapping

physical storage

PTDA

VMOB

va

va

VMOB

Va

VMOB

MTE

(pseudo defect)

hme

hme

boxnext

boxnext

va

va

va

va

va

va

va

va
The Page Frame Structure

Occasionally we need to enquire into the ownership and disposition of real storage. The PF is used to track the use of all frames of real storage, whether allocated, idle (pending freeing) or free.

The PF is formatted using the .MP KDB and DF command. .MP will optionally take the frame number (real address MOD 4K) as a parameter.

```
# .mp
ffe1b000 InUse: pVP=ff406000 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00000
ffe1b00c InUse: pVP=ff406050 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001
ffe1b018 InUse: pVP=ff40605a RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00002
ffe1b024 InUse: pVP=ff406064 RefCnt=0002 Flg=0 ll=00 sl=00 Blk=00000 Frame=00003
ffe1b030 InUse: pVP=ff40606e RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00004
```

Of particular interest are:

- **Frame=ffff**
  - The real storage frame number represented by this PF.

- **pVP**
  - The address of the related Virtual Page Structure (non-free PFs only). See next section.

- **ll**
  - The long term lock count.
  - This is non-zero when an otherwise non-resident page is long-term locked, that is prohibited from being discarded or swapped, and expected to be so for a relatively long time.

- **sl**
  - The short term lock count.
  - This is non-zero when an otherwise non-resident page is short-term locked, that is prohibited from being discarded or swapped, but temporarily so (much less than 10 seconds).

The Virtual Page Structure

VPs track the disposition of every page of virtual storage of every object. They enable the system to locate the data for the page, whether it is in RAM or on the Swap file.

VPs are formatted using the .MV KDB and DF command, which takes as a parameter the address of the VP, which may be obtained from the PF structure.

```
.mv %ff4060f0 15
VPI=0011 pVP=ff4060f0 Res Frame=0011 Flg=410 HobPg=0001 Hob=0009 Ref=001
VPI=0012 pVP=ff4060fa Res Frame=0012 Flg=410 HobPg=0002 Hob=0009 Ref=001
VPI=0013 pVP=ff406104 Res Frame=0013 Flg=410 HobPg=0003 Hob=0009 Ref=002
VPI=0014 pVP=ff40610e Swp Block=0001 Flg=0a0 HobPg=027b Hob=0026 Ref=001
VPI=0015 pVP=ff406118 Swp Block=0001 Flg=000 HobPg=0000 Hob=00e7 Ref=001
```

Of particular interest are:

- **Hob=nnnn**
  - The hob of the object to which this page belongs.

- **HobPg=nnnn**
  - The page number within the object.
Frame=ffff
The real storage frame number that backs this virtual page.

Block=bbbb
The Swap file 4K block that contains the data for this virtual page.

Page Management

The relationship between PF structures, Page Table Entries, Swap File Blocks and Memory Objects is shown in the following diagrams.

The relationship between PF structures, VP structures, Page Table Entries, Swap File Blocks and Memory Objects is shown in the following two diagrams

The first of these depicts the situation where storage is backed or committed by physical memory.
Backed Virtual Storage

physical storage

PF table

VP table

virtual storage

4K page

4K frame

VP frame

PTE

swapper

module

physical mapping

Page Management
The next diagram shows how this situation changes when storage is paged out.

Note that the Page Table Entry is used to record the swapper block number when the page is not present.
Page Management

Unbacked Virtual Storage

physical storage -> PF table -> VP table

4K frame <- PF -> VP

vpid

PTE -> swapper

physical mapping

module

virtual storage

4K page

VMOB -> VMAR

Ecb

$EcbP_g$
Aliasing

The situation described thus far can be further complicated by a technique known as aliasing. This is where one or more pages of an object may be mapped by page table manipulation to one or more pages of another object. In effect, this is partial object sharing.

This can occur between processes or within a process and is usually done for the following reasons:

- A device driver needs to create an I/O buffer to receive data at interrupt time and therefore in any context. The application that called the device driver also needs to have access to the results. This is commonly solved by the device driver making a UVIRT allocation in the system arena which aliases an application data buffer.

- A debugging application needs to access or even modify data and code of the debuggee. This is achieved through CS and DS selector aliasing.

- A Dos Virtual Machine needs to simulate the A20 line wrap-around. Storage addressed above the A20 line aliases to addresses module 2^20.

- A Dos Virtual Machine’s Private Arena address space is aliased in the system arena so that it may be accessed by Virtual Device Drivers in a context other than that of the VDM. The VDM handle (HVMD) is the alias address, which the VDD may add to any Private Arena Address to obtain a context independent access to a location in a given VDM.

These situations require the introduction of another memory management control block: the Alias Record (VMAL). Each VMAL has a unique handle or hal, which is the table entry from which the VMAL is allocated.

Where aliasing occurs, the handle to the alias record (hal) is saved in the VMAR of the aliasing address range.

In the case of memory aliasing the VMAL contains the handle to the PTDA of the aliasing process.

In the case of CS/DS aliasing within a process the VMAL contains the CS selector.

The link field of the VMAR is used to link together aliasing and aliased address ranges.

Alias records may be formatted using the .ML command as shown in the following example:

```ml
hal=0001 pal=%fc5de020 har=00af hptda=00ae pgoff=00000 f=001
hal=0002 pal=%fc5de028 har=00b0 hptda=00ae pgoff=00000 f=001
hal=0003 pal=%fc5de030 har=007a hptda=00ae pgoff=00000 f=001
hal=0004 pal=%fc5de038 har=0160 cs=00e6 ds=d446 cref=001 f=13
hal=0005 pal=%fc5de040 har=017f hptda=00ae pgoff=00000 f=001
hal=0006 pal=%fc5de048 har=0197 hptda=00ae pgoff=00000 f=021
hal=0007 pal=%fc5de050 har=0198 hptda=00ae pgoff=00000 f=021
hal=0008 pal=%fc5de058 har=0199 hptda=00ae pgoff=00000 f=021
hal=0009 pal=%fc5de060 har=01c8 hptda=00ae pgoff=00000 f=001
hal=000a pal=%fc5de068 har=01db cs=0056 ds=d446 cref=001 f=13
hal=000b pal=%fc5de070 har=020b cs=0056 ds=d446 cref=001 f=13
hal=000c pal=%fc5de078 har=0242 cs=0056 ds=d446 cref=001 f=13
```

CS Aliasing is depicted in the following diagram:
CS Alias of Shared Instance Data

Instance Data

Data Object

VMAR

VMOB

VMOB

VMOB

physical mapping

Physical Storage

CS Alias

VMAR

VMAL

Code Object

DS

hcb

hcb

hcb

hcb

va

link

va

bal

bar

bar

bar

hcbnext

hcbnext

physical mapping
The following diagram depicts multiple process memory aliasing:
Memory Aliases in Multiple Processes

VMAR → VMOB → VMAR → VMAR

hal → hco → link → link → va

VMLAL → PTDA

Data Object

Alias Data

Physical storage

physical mapping

physical mapping

R.M. 28th Aug 35 - wrashe
Who Owns Virtual Memory?

Given a virtual address, the procedure for determining who owns and is using this memory essentially amounts to the following steps:

1. If the question of ownership relates to a known process’s private storage then determine its hptda.
2. Locate the arena record(s) that encompass the address.
3. If more than one then select the one that relates to the process of interest (if known) by matching the hptda.
4. Locate the object records that are chained to the arena record.
5. If more than one then select the process of interest by matching the hptda.
6. Note the own and hmte values and their interpretation on the right-hand side of the VMOB display.
7. If necessary format the own and hmte VMOBs.
8. If either is an MTE then use .LM or .LMO, with the hob as parameter, to format the MTE.
9. If the memory is shared (hco=nnnnn appears in the arena record display) then format the chain of VMCOs and select the one that matches hptda from step 1.

Fortunately this task is reduced in complexity because of the M or match option that exists with both the .MO and .MA commands.

.MOM addr will display the VMOB of a pseudo-object that matches the addr if it exists. PTDAs are pseudo-objects and their addresses are listed by the .P command.

.MAM addr will search for all arena records whose address range encompasses addr. Under the kernel Debugger this search is restricted to the current context unless the A option (all contexts) is also specified. Under the dump formatter A in always in effect.

The C option further reduces the effort. This is the chain option and is applicable to .MO, .MA, .MC and .ML commands. Chaining formats all VMOBs, VMARs, VMCO and VMALs that are chained from each VMAR associated with the VM control block being formatted.

.MAMC (or .MAMAC under the DF) are the default options if just .M is specified. Furthermore the matching address defaults to the current CS:EIP.

The following sections illustrate memory ownership in:

Shared Arena Global Data
Shared Arena Instance Data
Private Arena Shared and Private Data
Physical Storage.

Further examples in memory management exploration, including looking at aliasing may be found in Exploring Memory Management.

Shared Global Data

Who owns %12123456?

#.m %12123456

*har par cpg va flg next prev link hash hob hal
0297 tfee2904 00000000 369 0312 0295 0000 009e 02a1 0000 hco=0057f
hob har hobxst flgs own hmte sown,cnt lt st xf
02a1 0297 0000 4a2d fff5 0302 0000 00 00 00 give
hco=057f pco=ffe7b96 hconext=%20241 hptda=06d1 f=1e pid=0059
hco=0241 pco=ffe6fb60 hconext=004ce hptda=04b2 f=1e pid=0043 c:pmspool.exe
This is shared arena global data because of the presence of the hco chain. The storage was dynamically allocated by PMMERGE.DLL as giveable storage. It is currently being referenced by 14 processes.

--------------------------------------------

Shared Instance Data

Who allocated %13fa1234?

%13fa1234 is the shared arena (there is no hptda associated with the arena record).

This is shared instance data because of the chain of related object records.

The storage was allocated by hmte=a3, but their are multiple owners.

mte a3 is DOSCALL1.DLL

This is shared instance data because of the chain of related object records.
Private Data

Who owns #17:0 in thread slots 8 and 9?

>> First find the hptda's for each of the slots of interest since we are
>> looking at private arena storage
#
.* p8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008 0008 0001 0008 0007 blk 0200 abd2f000 abe497f0 abe28bf0 01 PMSHL32
* .mom %abe497f0
   hob va flgs own hmte sown,cnt lt st xf
027a %abe497f0 8000 ffc9 ff79 0000 00 00 00 00 ptda 0008 c:pmshell.exe
#
.* p 9
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0009 0004 0001 0003 0001 blk 081f abd30000 abe48614 abe28de8 00 GAMBIT
* .mom %abe48614
   hob va flgs own hmte sown,cnt lt st xf
02ac %abe48614 8000 ffc9 02a8 0000 00 00 00 00 ptda 0004 c:gambit.exe

>> Next list all the owners of #17:0
#
.* #17:0
* har par cpg va flg next prev link hash hob hal
026d %feef2568 00000010 %00020000 1d9 029a 026c 0000 0000 029d 0000 hptda=02ad
hob har hbnxlt flgs own hmte sown,cnt lt st xf
029d 026d 0000 0838 029e 029e 0000 00 00 00 00 shared c:lanmsgexe.exe
* har par cpg va flg next prev link hash hob hal
0277 %feef2644 00000010 %00020000 1d9 0276 0272 0000 0000 02b0 0000 hptda=02ac
hob har hbnxlt flgs own hmte sown,cnt lt st xf
02b0 0277 0000 0838 02b1 02b1 0000 00 00 00 00 shared c:gambit.exe
* har par cpg va flg next prev link hash hob hal
02a0 %feef29ca 00000010 %00020000 179 02a4 02af 0000 0000 02e8 0000 hptda=02e1
hob har hbnxlt flgs own hmte sown,cnt lt st xf
02e8 02a0 0000 002c 02e7 02e7 0000 00 00 00 00 priv 0007 c:landll.exe
* har par cpg va flg next prev link hash hob hal
02aa %feef2a66 00000010 %00020000 179 02ab 02a9 0000 0000 02f8 0000 hptda=02fa
hob har hbnxlt flgs own hmte sown,cnt lt st xf
02f8 02aa 0000 002c 027a 027f 0000 00 00 00 00 priv 0008 c:pspshell.exe
* har par cpg va flg next prev link hash hob hal
02fc %feef31b2 00000010 %00020000 1d9 02fd 02fb 0000 0000 0360 0000 hptda=0359
hob har hbnxlt flgs own hmte sown,cnt lt st xf
0360 02fc 0000 0838 035f 035f 0000 00 00 00 00 shared c:harderr.exe
* har par cpg va flg next prev link hash hob hal
0360 %feef3a4a 00000010 %00020000 1d9 0361 035f 0000 0000 03d0 0000 hptda=03c9
hob har hbnxlt flgs own hmte sown,cnt lt st xf
03d0 0360 0000 0838 03cf 03cf 0000 00 00 00 00 shared c:ddaemon.exe
* har par cpg va flg next prev link hash hob hal
0360b %feef3b3c 00000010 %00020000 1d9 036c 036a 0000 0000 03e0 0000 hptda=03d9
hob har hbnxlt flgs own hmte sown,cnt lt st xf
03e0 036b 0000 0838 03df 03df 0000 00 00 00 00 shared c:pspdemon.exe
* har par cpg va flg next prev link hash hob hal
0378 %feef35ca 00000010 %00020000 1d9 0379 0377 0000 0000 03f3 0000 hptda=03ec
hob har hbnxlt flgs own hmte sown,cnt lt st xf
03f3 0378 0000 0838 03f2 03f2 0000 00 00 00 00 shared
* har par cpg va flg next prev link hash hob hal
040e %feef493e 00000010 %00020000 179 045c 040f 0000 0000 04c6 0000 hptda=04b2
hob har hbnxlt flgs own hmte sown,cnt lt st xf
04c6 040e 0000 002c 04b2 0522 0000 00 00 00 00 priv 0043 c:pmspool.exe
This is private arena data of some sort, whose address range is present in 13 processes.

The hptda for pid 4 (slot 9 is 2ac)

The second major entry from .m output (har=277, hptda=2ac) is for gambit.exe in pid 4.

The owner and hmte are the same (2b1). This indicates a code segment within the module gambit.exe.

.LMO 2b1 show this to be in segment 2 of gambit.exe

The storage in pid 8 (slot 8) is shown in the 4th entry, har=2aa.

Here own=27a and hmte=2f7.

The owner is shown to the right of the VMOB as being pid 8. We can check this by displaying hob 27a. This turns out to be a ptda for pid 8, as we saw when we used .mom against the PTDA address.

.lmo 2f7 shows this to be the MTE for pmshell.exe. We concluded that pmshell has allocated private memory in pid 8 at this address.

------------------------------------------------------

Physical Memory
Who owns physical address %%90123?

Physical address %%90123 is in frame 90.

This is currently assigned to VP at %ff408576 and is on the swap file at block 272.

The VP tells us the hob and page within the hob.

.MOC will format the VMOB and associated VMAR.

We can check that this is correct from the page table entries for the 17th (0x11th) page of the object's virtual address.

Thread Scheduling and Dispatching Topics

Part 2 of our discussion on The Wait Condition centres on the Scheduler and Dispatcher and the mechanisms that govern when threads will or will not run.

This is considered from the perspective the system, which leads us to divide the discussion into two cases:

- **Blocking - Voluntary Suspension**
- **Involuntary Suspension**

Blocking - Voluntary Suspension

We now turn our attention to blocking, which is mechanism that threads use to give up processor time voluntarily to wait for an event to occur or a resource to become available.

The term voluntary is chosen from the perspective of the scheduler and not necessarily from the application's perspective. In this context voluntary suspension refers to an action taken by a thread to give up its time-slice. This will include direct actions such as waiting on semaphores as well as calling APIs, which for internal reasons need to wait for a resource or an event.

PROCBLOCK and its counterpart PROCRUN are the two kernel routines at the heart of the block/run mechanism. These are callable directly by kernel component and also by Device Drivers and File System Drivers through a small interface layer. Application code only gets to call PROCBLOCK and PROCRUN indirectly through system APIs and in particular through the semaphore APIs.

The block/run mechanism is designed with the following criteria:

- A thread should be able to block without the waking thread having to know whether anyone, or who, had blocked on a resource
Multiple threads should be able to wake when an event or resource becomes available. This is achieved by having an abstract token, known as the Block ID, associated with the resource or event. The BlockId is passed to PROCBLOCK when a thread blocks. Similarly when another thread wishes to wake threads waiting for a resource or event the BlockId that represents the resource or event is passed to PROCRUN.

In addition to the BlockId, callers of PROCRUN receive a flag that indicates whether all or just the highest priority thread waiting on the BlockId should wake.

This mechanism has shortcomings unless certain constraints are applied:

- BlockIds need to be subject to a convention that gives uniqueness otherwise it is possible that threads will incorrectly block and run. A solution is to use the address of a control block memory object that relates uniquely to the resource or event.
- If addresses are to be used for BlockIds then they must point to global data for reasons of uniqueness. Furthermore, if they are to be reference by disabled code then the storage needs to be in resident memory. This more or less implies that addresses must be taken from within the System Arena.
- If BlockIds are in use that do not represent addresses then they must not conflict with any potential addresses used as BlockIds.
- Even if addresses are used there is no accounting information that says who owns the related resource.

A workable scheme is implemented by limiting the direct use of PROCBLOCK and PROCRUN to system code, device drivers and file system drivers, all of which have access to the System Arena.

Apart from three special conventions the system and most device drivers use addresses as BlockIds. There are three system defined conventional BlockIds are:

- fffe.... results from a RAMSEM wait.
- fffd.... results from a MUXWAIT.
- ffca.... results from a Child Wait
- x....... (x=a - f)
  Linear address of the memory object of control block that relates to the resource.
- ........ Probably selector:offset address of the memory object or control block that relates to the resource.

This scheme could be subverted by device drivers, but in general they will choose to block on addresses of resources they own, which are usually allocated out of the system arena and addressed using a GDT select:offset.

Accountability remains an exposure. For BlockIds that are addresses the owner of the memory that the BlockId points to gives a big clue. For conventional BlockIds we have to do more work. These are discussed in detail later. We will first we look at an example of a BlockId that is an address.

**Basic Technique:**

The technique for analysing blocked threads is two-pronged:

1. We can look at the wait from the application perspective by examining the current user registers and by trying to identify the API issued. This is usually relatively easy but often gives no clue as to the underlying wait since any single API may block on many occasions for many reasons.
2. Examine the problem from the Internal, or Kernel perspective to determine what an API might be waiting for. This process starts with finding the associated BlockId.

When a thread blocks its BlockId is stored in the TCB TCBSleepld field. Conveniently, this is formatted by using the .PB KDB and DF command.

**Note:**

.PB under DF lists non-blocked threads. BlockIds are irrelevant for such threads.

.PB also attempts to interpret the BlockId. The full details of these are given in the Kernel Debugger and Dump Formatter Command Reference. In addition to classifying the BlockId, .PB examines TCB_SemInfo and TCB_SemDebugAddr.

For many semaphore originated BlockIds TCB_SemInfo is used to store the address or handle of the user's semaphore that lead to the
thread blocking. .PB will attempt to locate a near symbol to the semaphore address and display it.

Under the kernel Debugger, TCB_SemDebugAddr is used to store the address of the caller to the Semaphore API when the thread blocked. If this field is not 0xffffffff .PB attempts to locate a near symbol to the caller and display it.

Once we have the BlockId, TCB_SemInfo, and TCB_SemDebugAddr we are able to begin searching for information associated with reason for blocking.

The next step is to decide whether the BlockId is one of the three special categories or to be treated as an address.

-----------------------------

Blocking on the Address of a Resource

The initial analysis of BlockIds that are linear addresses uses the .M command to determine ownership.

If we have appropriate symbols loaded, LN against the BlockId can also be very informative.

As mentioned in the previous section, for addresses to be effective BlockIds they must be unique and so are generally allocated from the system arena. Most allocations from the system arena are 'labelled' with a system object Id. If the .MO command is used against a system object Id it will display a meaningful mnemonic for the Owner Id. In many cases the Mnemonic is for a system control block or buffer.

BlockIds that address the beginning of a control block tend to be used for serialising updates to the control block. There may be processes that a control block is associated with. These are often serialised by using the address of a field within the control, that is specifically associated with the process.

A complete list of system object Ids may be found in the under the Kernel Dubugger Command Reference under the .MO command description.

We now look at some examples:

**File System - Device Driver**

```
# .pb41
Slot  Sta BlockID  Type  Addr  Symbol
0041  blk  04085ca7 DEMO1
# ln 408:5ca7
No Symbols Found
# .m 408:5ca7
*har  par  cpg  va  flg  next  prev  link  hash  hob  hal
0079 kfee2fa70 00000010 kbf27000 129 0078 0077 0000 0000 007b 0000  sel=0408
hob  har  hbnst  flgs  own  hnte  sown  cnt  lt  st  xf
007b 0000 0000 0000 0000 sft=408
# .d sft 408:5ca7
sf_ref_count: 00000000  sf_hmode: 0092
sf_usercnt: 00000000  sf_hVPB: 0000
reserved: 00000000  sfctime: 0000
sf_flags(2): 00000000  sfi_cdate: 0000
sf_devptr: 00000000  sfi_atime: 0000
sf_FSC: 00000000  sfi_ddate: 0000
sf_chain: 00000000  sfi_mtime: 0000
sf_NFT: ffffffff  sfi_mdate: 0000
sfdFAT_firF1Eclus: 0000  sfi_position: 00000000
sfdFAT_clupos: 0000  sfi_size: 00000000
sfdFAT_lstclus: 0000  sfdFAT_clupos: 0000
sfdFAT_dirsec: 00000000  sfdFAT_lstclus: 0000
sfdFAT_dirpos: 00  sfdFAT_dirsec: 00000000
sfdFAT_name: DEMODEV2  sfdFAT_dirpos: 00
sfdFAT_EAHhandle: 00  sfdFAT_name: DEMODEV2
sfdFAT_PipeSfn: 00  sfdFAT_EAHhandle: 00
sfdFAT_HName: DEMO1  sfdFAT_PipeSfn: 00
sfdFAT_mode: 00  sfdFAT_HName: DEMO1
sfdFAT_size: 00  sfdFAT_mode: 00
sfdFAT_num: 00  sfdFAT_size: 00
sfdFAT_atime: 00  sfdFAT_num: 00
sfdFAT FileMode: 00  sfdFAT_atime: 00
sfdFAT_host: 00  sfdFAT FileMode: 00
sfdFAT_file: 00  sfdFAT_host: 00
sfdFAT_filetype: 00  sfdFAT_file: 00
sfdFAT_datalen: 00  sfdFAT_filetype: 00
sfdFAT_datalen: 00  sfdFAT_datalen: 00
sfdFAT_datalen: 00  sfdFAT_datalen: 00
sfdFAT_datalen: 00  sfdFAT_datalen: 00
sfdFAT_datalen: 00  sfdFAT_datalen: 00

# .p41
Slot  Pid  Ppid Csid Ord  Sta  Pri  pTSD  pPTDA  pTCB  Disp  SG  Name
0041  0012 000f 0012 0001 blk 0300 7bd19000 7bdfc218 67ddd568 00b0 c1 DEMO1
# .s41
Current slot number: 0041
# .r
eax=00000000 ebx=00000002 ecx=00000000 edx=0d3409ea esi=d02f0021 edi=00000009ea
Slot 41 is waiting on BlockId 04085ca7. This is too low to be a linear address. We assume selector:offset.

.M 408:5ca7 reveals the owner to be sft. This is a System File Table structure.

The .D command will format STFs, so we do so using the BlockId as the SFT address.

This SFT represents a device driver called DEMODEV2. We can tell because there is no MFT pointer in the SFT and the flags indicate a device.

From the application side we unassemble back from the CS:IP.

The application has just issued a call-gate instruction.

Examination of the call-gate GDT descriptor show we were calling DOSCLOSE in the kernel.

We are waiting for the close to complete, possibly the device driver has not returned completion status to the last I/O request.

Named Pipes

# .s 18
Current slot number: 0018
# .pb#
Slot  Sta BlockID Name Type Addr Symbol
0018# blk 06700012 EPWPSI
#
# .m 670:12
*har par cpg va flg next prev link hash hob hal
00a8 %fef1fe7a 00000010 %7b563000 129 00a7 00a9 0000 0000 00b4 0000 sel=0670
hob har hobnat flgs own hmte sown cnt lt st xf
00b4 00a8 0000 0124 ff31 0000 00 00 00 00 npipenp
# .p#
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0018# 000c 0000 000c 0001 blk 0200 7bc70000 7bd79964 7bd5b5f0 0ec8 00 EPWPSI
# .r
eax=00000000 ebx=00005552 ecx=00050000 edx=0000f020 esi=00000446 edi=0000302
elip=00014bb esp=000063da ebp=000063de iopl=2 -- -- nv up ei pl nz na po nc
cs=00ff ss=0000 ds=00ff es=00ff fs=150b gs=0000 cr2=00000000 cr3=001ac000
d01f:000014bb c9 leave
# u.cs:ip-10
Expression error
# u.cs:ip-10
d01f:000014ab c9 leave
d01f:000014ac caa000 retf 000a
DOSCALL1 DOSCONNECTNMPPIPE:
d01f:000014af c8000000 enter 0000,00
d01f:000014b3 ff7606 push word ptr [bp+06]
d01f:000014b6 9a0000131c call lcl13:0000

In this example the BlockId is 06700012. This is unlikely to be a linear address. We assume that it is 670:12.

.M 670:12 shows the owner to be npipenm. This is a named pipe name segment. Could the process be waiting for a pipe connection?

Looking at the application side we see that the last ring 3 instruction to be executed was a call-gate, which turns out to be DOSCONNECTNMPIPE in the Kernel.

These last two examples were reasonably revealing. More often than not we use .M against a BlockId (and other system data) and we get one of:

vmkshrw
vmkshro
vmkrhrw
vmkrhro

These are, so called Public Kernel Heaps. Fortunately each allocated heap block is imbedded in a structure that reveals the owner of the block. This is discussed next.

Kernel Public Heaps

The kernel has 7 heaps for general use by itself, device drivers and file system drivers. They have the following object id mnemonics:

vmkshro Swappable read-only heap
vmkshrw Swappable read/write heap
vmkrhro Resident read-only heap
vmkrhrw Resident read/write heap
krhr01m Resident read-only < 1Mb heap
krhrw1m Resident read/write < 1Mb heap
kbdsym Resident kernel debugger symbol heap

Not all heaps are always built. Note in particular:

hbdsym is not present under the RETAIL kernel.
vmkshrw is used for the krhr01m, krhrw1m, vmkshrw and vmkshro heaps under the RETAIL kernel.
vmkrhrw is used for both vmkhrw and vmkrhro under the RETAIL kernel.

Notice that each of the heaps is either resident or swappable.

Each heap is partitioned into blocks.

Swappable heap blocks have an 8 byte prefix followed by the block data.
Resident heap blocks have two forms:

- **Regular**: for smaller allocation. These have a 4 byte prefix.
- **Attributed or extended**: These use a 4 byte prefix and an 8 byte suffix.

---

### Swappable Heap Blocks

Kernel swappable heap blocks for allocated blocks have the following layout:

```
<size><owner><selector><data>
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size in bytes</td>
<td>63-32</td>
<td>Size of the block including the header in bytes ORed with signature 0x52000000.</td>
</tr>
<tr>
<td>owner</td>
<td>31-16</td>
<td>Owner of heap block. This is either a system owner (value between 0xff2d and 0xfff8, or a memory handle/pseudo handle such as an MTE pseudo-handle.</td>
</tr>
<tr>
<td>selector</td>
<td>15-0</td>
<td>GDT selector mapping block's data else null.</td>
</tr>
</tbody>
</table>

---

### Finding the owner of a Swappable Head Selector

```
# .m 8f0:0
*har par cpg va flg next prev link hash hob hal
0021 %fef1f2e0 00001400 %fca5f000 121 0022 0002 0000 0020 0022 0000 =0000
hob har hobxnt flgs own hmte sowm,cnt 1t st xf
0022 0021 0000 0225 ffef 0000 0000 00 04 00 00 vmkshrw
# d1 8f0
GDT
08f0 Code Bas=fca95000 Lim=00008ed3 DPL=0 P RE A
# dd %fca95000-10
%fca94ff0 00000000 00000000 52008ee0 08f0ff49
%fca95000 08e8b81e 32b8d88e 16ca1f00 06c89000
%fca95010 1e560000 e09e88b8 a23e83d8 06740009
%fca95020 eb63a5e8 c02b9003 0bfe4689 e90374c0
%fca95030 468b017e 10568b0e 52000805 6aff6a50
%fca95040 13969aff 5f3d1000 c4e77400 83260e5e
%fca95050 74000e7f 0142e903 0c47ff26 261276c4
%fca95060 2616448b 8918548b 5689fa46 0e468bfc
# .mo ff49
ff49 fsd2
# .ml
hmte=0982 pmte=fee0e1a4 mflags=0408b186 e:\ibmlan\netlib\splla.dll
hmte=097e pmte=fee0e1a5 mflags=0408b186 e:\ibmlan\netlib\lrml.dll
hmte=0979 pmte=fee0e1bac mflags=0408b186 e:\ibmlan\netlib\lrsrl.dll
hmte=096b pmte=fee0e1d60 mflags=0408b186 e:\iblan\netlib\netibm.dll
hmte=0164 pmte=fee0e2cc40 mflags=0498b1c8 e:\os2\dll\symon.fon
· · ·
hmte=0181 pmte=fee0e2cc80 mflags=4498bd5 e:\os2\dll\smtm.dll
hmte=031b pmte=fee0e2af80 mflags=0428a1c9 e:\ibmlan\netprox\netwksta.200
hmte=030e pmte=fee0e59f90 mflags=0428a1c9 e:\netware\nwifs.ifs
hmte=0160 pmte=fee0e1ff4c mflags=0428a1c9 d:\dataex2\ufsfsd2.ifs
hmte=0117 pmte=fee0f5df60 mflags=0428a1c9 e:\os2\cdfs.ifs
hmte=00d2 pmte=fee0f53990 mflags=0428a1c9 e:\os2\hpfs.ifs
```
We use .M to find that the owner of 8f0:0 is vmkshw.

So, we look at the descriptor for 8f0 to find it's base address. Note that the selectors assigned to kernel heap blocks address the data portion only.

We dump out 0x10 bytes before the selector base to show the block header to be 0x52008ee0 0x08f0ff49. This tells us the length of the block including header is 8ee0. (Data sizes are rounded up to the next quad-word). The user of the block is ff49.

Note:

The following short cut could have been used:

dd %(8f0:0)-10

.MO ff49 shows fsd2. This is the second file system driver to initialise.

.LML will list DLLs, Fonts, and FSDs, newest first. Counting back from the end we see FSD1 is HPFS and FSD2 is CDFS.

.LMO 117 confirms that 8f0:0 does belong to CDFS.IFS.

--------------------------------------------

Resident Heap Blocks

Kernel resident heap blocks are of two types, regular and attributed.

Regular blocks are the simplest and most common type. They have the form:

<simple header><data>

<simple header> is a dword (32-bits) having the following layout

<owner><prev block free flag><size in dwords><yielded flag><type flag>

Field     Bits            Description
----------  ------------------------------
owner      31-16           Owner of heap block. This is either a system owner (value between 0xff2d and 0xfff8, or a memory handle/pseudo handle such as an MTE pseudo-handle.
prev block 15              1 if previous block is free, else 0
free flag
size in dwords 14-2        Size of the block including the header in dwords.
yielded flag 1              1 if a free block search yielded the CPU while looking at this block, else 0
type flag    0               0 (indicates Regular Block)

Extended blocks contain a two-part header and have the following form:

<size header><data><header extension>

<size header> is a dword (32-bits) having the following layout

<extra flags><size in dwords><yielded flag><type flag>.

Field     Bits            Description
extra flags    31-24    Additional flags.
    Bit 31 - set if block is free
    Bit 30 - set if prev block is free
    Bits 29-24 - reserved and 0
size in dwords 23-2    Size of the block including the
yielded flag    1    1 if a free block search yielded
    the CPU while looking at this block,
    else 0
type flag       0    1 (indicates Extended Block)

<data> is the data area available for use by the client and is
always dword granular and dword aligned.

<header extension> is a dword-granular structure containing
the following information

<owner><selector><hmte><pad>

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>63-48</td>
<td>Owner of heap block. This is either a system owner</td>
</tr>
<tr>
<td>selector</td>
<td>47-32</td>
<td>GDT selector mapping block's data</td>
</tr>
<tr>
<td>hmte</td>
<td>31-16</td>
<td>hmte associated with this heap block?</td>
</tr>
<tr>
<td>pad</td>
<td>15-0</td>
<td>padding for double word alignment</td>
</tr>
</tbody>
</table>

When a block is free, its data portion contains additional information. The first two dwords contain forward and backward pointers to the next
and previous blocks on the free list. The last dword contains a copy of the previous block pointer. Note that extended free blocks do not
have an owner field, so bit 31 of their header is set indicating that they are free.

The hmte field of the header extension is no longer used for any specific purpose.

**Now for an example of a regular heap block.**

```plaintext
# .s 47
Current slot number: 0047
# .pb #
Slot Sta BlockID Name Type Addr Symbol
0047# blk fe04c8e8 PMSHL32
# .m %fe04c8e8
*har par cpg va flg next prev link hash hob hal
0003 %fef1f04c 00001000 %fdf1f000 001 0002 0020 0000 0000 0003 0000 =0000
hob par hobnxt flgs own hmte sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 0000 00 06 00 00 vmkrhrw
# dd %fe04c8e8-10
%fe04c8d8 00031c3f 00000000 00000000 ffc20010
%fe04c8e8 00000010 00000000 00000000 ffc20010
%fe04c8f8 00000010 00000000 00000000 ffe900a8
%fe04c908 fe0c767c fe0c0ee0 00000000 00000000
%fe04c918 00000000 00000000 00000000 00000000
%fe04c928 00000000 00000000 00000000 00000000
%fe04c938 00000000 00000000 00000000 00000000
%fe04c948 00000000 00000000 00000000 00000000
# .mo ffc2
ffc2 semstruc
# .d sem32 %fe04c8e8
```

Type: Private Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 0010fe04
In this example we are interested in slot 47. Its BlockId is owned by \textit{vmkrhrw}.

We dump the heap block from 0x10 bytes before the start.

Note that the low order bit of the header is 0, therefore a regular block.

Since the two low order bit are flags and the size is in double words we conveniently ignore these to obtain the size in bytes, which happens to be 0x10.

The block is owned by ffc2, which .mo tells us is \textit{semstruc}.

This is very good news because all the semstruc owner relates to the 32-bit semaphore APIs. The .D command formats these for us.

Finally note that if we attempt to look at this from the application perspective we see from .P that the TSD is swapped out (Disp is blank). This means that the user registers for slot 47 can't be loaded. Furthermore attempts to look at the registers are unpredictable as DF and KDB will have not changed the values since they last loaded registers.

This is a case where BlockId analysis will give us a clue even if the application data is unavailable.

**Lastly we look at an extended heap block.**

```
# .s 4b
Current slot number: 004b

# .pb#
Slot Sta BlockID Name Type Addr Symbol
004b# blk 21a0ade0 WKSTAHLP

# .m 21a0:ade0
*har par cpg va flg next prev link hash hob hal
0003 %fef1f04c 00001000 %fdf1f000 001 0002 0020 0000 0000 0003 0000 =0000
hob har hobnext flgs own hmte sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 00 06 00 00 vmkrhrw

# dg 21a0
21a0 Code Bas=fe070000 Lim=0000bd5b DPL=0 P RE A
# dd %fe070000-10
%fe06ff0 00000000 00000000 fe06fd72 4000bd6d
%fe070000 10d0006b 9090cbcb 9090cb90 000af390
%fe070010 3ce00104 6aee8b55 8d26a01 16eb866
%fe070020 6aee8b55 8d26a02 0aeb868 6aee8b55
%fe070030 8d26a00 46c6d866 561e00ee 21906857
%fe070040 1e7ec41f 7145568b 750001c2 56b520e
%fe070050 27e2830e 29558826 250e5eb5a 000af390
%fe070060 4000c2f7 c0330574 f70d39e9 7480000c2

# dd %fe070000-4+bd6c-10
%fe07bd58 fee2e9de 00000000 ff4c21a0 fdf1ff32
%fe07bd68 fe9008c fe06fd70 fe02a470 00000000
%fe07bd78 00000000 00000000 00000000 00000000
%fe07bd88 00000000 00000000 00000000 00000000
%fe07bd98 00000000 00000000 00000000 00000000
%fe07bdab 00000000 00000000 00000000 00000000
%fe07bdc8 00000000 00000000 00000000 02010000
%fe07bdcb 00000000 00000000 00000000 00000000

# .mo ff4c
ff4c fsd5

# .1ml
htmte=0982 pmte=ffe01a14 mflags=0408b186 e:\ibmlan\netlib\splla.dll
htmte=097e pmte=ffe01a54 mflags=0408b186 e:\ibmlan\netlib\shrml.dll
htmte=097a pmte=ffe01b8c mflags=0408b186 e:\ibmlan\netlib\lnrsl.dll
htmte=096b pmte=ffe01d60 mflags=0408b186 e:\ibmlan\netlib\netibm.dll
htmte=0164 pmte=ffe02cc40 mflags=0498b1c8 e:\os2\dll\sysmono.fon
```

What does BlockId 0x21a0ade0 represent?

We assume selector:offset and discover the owner is vmkshrw.

We dump the descriptor for selector 21a0 to find its base address.

Next we dump 0x10 bytes before the descriptor base to see the heap block header.

In this example the low order bit of the header is 1 so we have to look at the header extension for the owner information.

Adding the length to the base and backing off 0x10 bytes again we uncover the block header extension.

Note:

The following short cut could have been used:

```
dd %((21a0:0)-4+bd6c-10
```

In this case the owner is ff4c or fsd5. This is the 5th FSD to initialise.

We list the FSDs by using .LML and pick the 5th from the bottom. This turns out to be netwksta.200.

-----------------------------

Blocking on a ChildWait

When a process calls DosWaitChild and blocks waiting for a child process to terminate, the BlockId is of the form:

```
ffcapppp where pppp is the process id of the waiting thread.
```

The BlockId doesn't help us pin-point the processes being waited for.

All the child process have to be examined. The process status byte at offset 0xa into the Local information segment has either of the following bits set if the parent cares about termination of the child:

0x10 The parent cares

0x20 The parent did an exec-and-wait

The local information segment is embedded in the PTDA at the following offsets:
Blocking on a RAMSEM

Potentially this is the most problematical type of wait to deal with. The BlockId is conventional and of the form:

```
fffexxxx where xxxx is taken from the low order word of the user's RAMSEM.
```

There is no accountability associated with this type of semaphore. It is the responsibility of the user to manage their own accounting information. Accordingly most applications tend to imbed RAMSEMs into larger structures, which contain information such as use counts, owner identification, timeouts.

Two structures in particular are in common use:

- The Fast Safe RAMSEM
- The PM Fast Safe RAMSEM

The first step with RAMSEM BlockIds is to locate the user’s RAMSEM address.

Next check ownership just in case this gives a clue to the associated process.

Ownership is indicated by a non-zero value in byte 0 of the RAMSEM. Very occasionally a RAMSEM is owned by the system. When this happens the ownership flags takes the value of the owning process.

We hope that the RAMSEM is embedded in either a Fast Safe RAMSEM or PM Fast Safe RAMSEM.

Both of these structures have a length prefix. The PM version is 0x12 and the non-PM version 0x0e.

Display storage before the RAMSEM and examine offset -0x12. Is this word 0x0012? If not then this is not a PM FSRS. Try -0xe. Does that contain 0x000e? If not then we will have to resort to more speculative analysis.

If either of these lengths correspond, look at the next two words, these contain the owning PID and TID. See whether this process and thread exists and what it is doing.

**Note:**

Tid is sometimes 0 when there is only one thread in a process.

If this technique fails us then check the owner of the semaphore address, which is saved in TCB_SemInfo and displayed by the .PB command. The owner of the semaphore, if it has not died, has to one of its accessors. If the RAMSEM is located in a Private Arena, then the owner is limited to one of the threads of the process that has blocked. If it is in shared storage, then the owned will be a thread in one of the processes on the VMCO chain. If we are lucky, the number of possibilities will be small, though this is not guaranteed.

The following example illustrates this technique:

```plaintext
>>> Slot 31 is blocked. Why?
```

```
Slot  Sta BlockID  Name     Type        Addr    Symbol
0031  blk fffe01ba aires  RamSem   e69f:000a
```

---
>>> Bad news! a RamSem. First check to see if its imbedded in a
>>> FastSafeRamSem. We look at the RamSem address, back a few bytes

% cmd

>>> Pid 19, Tid=0 (this is OK if just one thread in process 19).

---
>>> Pid 19 is single threaded and is blocked. See what its Block-Id is.

##.pb 33
Slot Sta BlockID Name     Type        Addr    Symbol
0033 blk fffe01bb orian  RamSem    e66f:0000

>>> Once again a RamSem. This time there's no point in looking back
>>> a few bytes to see if it's imbedded in a FastSafeRamSem because
>>> the RamSem is allocated at the beginning of segment e66f.

>>> Our only hope is to see who else has access to this semaphore.

##.m e66f:0000
*har     par      cpg        va    flg next prev link hash hob  hal
0420 %fed03aca 00000010 %1ccd0000 369 03f1 0075 0000 001b 0000  hco=007ff
hob  har hobnat flgs own hme sown,cnt lt st xf
051b  0420 0000  4a2c ff82 04f1  0000 00  00 00 00 mshare
hco=007ff pco=fe85f816 hconext=007b6 hptda=04d1 f=16  pid=0019 a:orian.exe
hco=007b6 pco=fe85f6a9 hconext=00000 hptda=04fd f=17  pid=0018 a:aries.exe

>>> The RamSem is allocated in Named Shared Storage (mshare is the
>>> owner). The only two processes able to access this are Pids 19 and
>>> 18. Pid 19 is this thread, which we know doesn't own this RamSem
>>> since it's waiting for it. The leaves 18.

>>> We can't be certain from the evidence presented so far, but we can
>>> say:
>>> Either the RamSem is owned by 18, or it was owned by another
>>> thread that has since terminated. If it is owned by 18 then we
>>> have a deadlock between 18 and 19:
>>> orian.exe owns the FSRamSem and is waiting for the RamSem.
>>> aries.exe owns the RamSem and is waiting for the FSRamSem.

Fortunately simple RAMSEMs are becoming something of the past. And now that PM is 32-bit we will not see many Fast Safe RAMSEM
either. We will look in detail later on at the semaphore structure that has replaced the FSRSEM in PM: the PMSEM and GRESEM.

--------------------------------------------

The MUX Wait

The last category of BlockIds to consider is the MUXWAIT. This has a BlockId of the form:

fffdssss       where ssss is the slot id of the waiting thread.

A MUXWAIT is a multiplex semaphore wait. The semaphore comprising the MUX list may be:

RAMSEMs
SYSSEMs
32-bit Event & Mutex SEMs

We will consider each of these in turn.

The first step is to format the muxtable. This comprises 9-byte entries. +0x2 is the slot number of the waiter. +0x5 indicates the type of
semaphore. +5 is the semaphore handle, which is interpreted according to type as follows:

0x00 Entry unused
0x01 handle is offset of SYSSEM from selector 400
0x02 Entry is a hob:offset of RAMSEM
The following shows an example formatted MUXTABLE. There are up to 255 entries, but in practise the entries in use are grouped at the beginning of the table.

# db muxtable+(9*0) l9
0400:000048be c7 48 14 00 02 1a 07 be-00                      GH.....>.n
# db muxtable+(9*1) l9
0400:000048c7 d0 48 15 00 02 5c 07 be-00                      PH...\.z.
# db muxtable+(9*2) l9
0400:000048d0 ff ff 15 00 02 64 07 fa-03                      .....x.z.
# db muxtable+(9*3) l9
0400:000048d9 e2 48 1f 00 02 58 07 fa-03                      bH...X.z.
# db muxtable+(9*4) l9
0400:000048e2 fd 48 1f 00 02 5c 07 fa-03                      }H...\.z.
# db muxtable+(9*5) l9
0400:000048eb c3 49 1f 00 02 50 07 fa-03                      CI...P.z.
# db muxtable+(9*6) l9
0400:000048e4 57 49 58 00 02 61 01 64-07                      WIX.\ad.
# db muxtable+(9*7) l9
0400:000048e6 06 49 1f 00 02 60 07 fa-03                      .I...'.z.
# db muxtable+(9*8) l9
0400:000048e8 0f 49 1f 00 02 64 07 fa-03                      .I...\d.z.
# db muxtable+(9*9) l9
0400:000048eb c7 49 58 00 02 61 01 64-07                      WIX.\ad.
# db muxtable+(9*a) l9
0400:000048eb 06 49 1f 00 02 60 07 fa-03                      .I...'.z.
# db muxtable+(9*b) l9
0400:000048e8 0f 49 1f 00 02 64 07 fa-03                      .I...\d.z.
# db muxtable+(9*c) l9
0400:000048eb 06 49 1f 00 02 60 07 fa-03                      .I...'.z.
# db muxtable+(9*d) l9
0400:000048e8 0f 49 1f 00 02 64 07 fa-03                      .I...\d.z.
# db muxtable+(9*e) l9
0400:000048eb 06 49 1f 00 02 60 07 fa-03                      .I...'.z.
# db muxtable+(9*f) l9
0400:000048e8 0f 49 1f 00 02 64 07 fa-03                      .I...\d.z.
# db muxtable+(9*10) l9
0400:00004921 2a 49 58 00 01 f0 53 00-00                      *IX..pS..
In this example there are only semaphore types 0, 1, 2 and 3. We will illustrate unravelling each of these in turn. For type 4 see the later section on 32-Bit Semaphores.

The SYSSESEM

The SYSSESEM block id points to a SYSSESEM Table Entry.

Note:

In a MUXWAIT only the offset is recorded in the MUX table entry. This should be used with selector 400.

In a single SYSSESEM, the BlockId is the selector:offset to the SYSSESEM Table Entry. The .PB command will display the SYSSESEM name.

The example below is from a MUXWAIT which includes a SYSSESEM

>> The MUXTABLE entry for slot 58. SYSSESEM offset = 53f0

```
# db muxtable+(9*b) l9
0400:00004921 2a 49 58 00 01 f0 53 00-00                      *IX..pS..
```

```
# .p 58
Slot Pid Ppid CsId Ord Sta Pri  pTSD  pPTDA  pTCB  Disp SG Name
0058 0014 0000 0014 0004 blk 021f 7bd30000 7bdfd260 7bde23f0 0eac 10 WKSTA
```

```
# .pb 58
Slot Sta BlockID  Name     Type        Addr        Symbol
0058 blk fffd0058 WKSTA    MuxWait
```

>> The SYSSESEM Data Table Entry

>> slot = 0058

>> flag = 02

```
>> 01= waiting
>> 02= mux waiting
>> 04= owner (pid/tid) died
>> 08= exclusive syssem
>> 10= name entry needs removing
>> 20= tid owner died
>> 40= exit list thread owns this sem
```

>> reference count = 01

>> request count (by this owner) = 0

>> padding=00

```
# db 400:53f0 16
0400:000053f0 58 00 02 01 00 00                               X......
```

>> SYSSESEM names are stored in a Record Management Package (RMP)

>> whose selector is the high word of:

```
# dd syssesemrmpdhdl l1
0400:0000595a 04d00004
```

>> The RMP has a 0x14 byte header followed by variable length entries.

>> Each entry is prefixed with a word length followed by the entry data.

>> The entry data is the word offset of the corresponding SYSSESEM Data Table

>> followed by offset 2 - n of the semaphore name. (the offset overlays

>> the first two bytes of the name which are always 'SE').

>> Scan the table looking for entry with offset 53f0...

```
# db d0:0
04d0:00000000 00 06 d0 02 0d 01 5b 03-01 00 00 00 04 00 00 ..P...[........
04d0:00000010 36 ff 00 00 10 00 5a 53-45 44 5e 56 49 4f 50 4e 6....SEM\VIOPO
04d0:00000020 50 55 50 00 10 00 60 53-45 44 5e 56 49 4f 50 52 PUP...SEM\VIOPO
04d0:00000030 54 53 43 00 12 00 66 53-45 44 5e 44 41 54 41 45 TSC...fSEM\DATAE
04d0:00000040 58 2e 45 52 50 00 0b 00-00 6c 53 45 44 5e 54 44 41 X.ERR...LSEM\DAT
04d0:00000050 41 45 58 2e 4c 4f 47 00-14 00 72 53 45 4d 5c 49 AEX\LOG...rSEM\I
```
**The MUX RAMSEM**

In a MUX the MUX table entry:

```
>> slot = 1f, type = 2, hob= 03fa, offset=0758
```

In this example we look at the MUXSEM being waited on by slot 1f.

```
>> The MUX table entry:
>> slot = 1f, type = 2, hob= 03fa, offset=0758
```

```
0400:000048d9 e2 48 1f 00 02 58 07 fa 03 bH...X.z.
# db muxtable+(9*4) l9
```

```
>> Use .MOC to find the linear address
```

```
# .moc 3fa
*har     par      cpg        va    flg next prev link hash hob   hal
039b %fef23f5c 00000010 %1a350000 379 0413 039c 0000 0000 03fa 0000  hco=00c45
hob   har hobnxt flgs own  hmte  sown,cnt lt st xf
03fa  039b 0000  082c 03fb 03fb  0000 00  00 00 00 shared    e:pmshapi.dll
hco=0c45 pco=ffe77d74  hconext=00e0 hptda=0873 f=16 pid=00e0 e:cmd.exe
```

```
>> This is owned by PMSHAPI. LN gives a label.
```

```
# ln %1a350750
%1a350750 PMSHAPI ASEMRS + 8
```

```
--------------------------------------------
```

```
> We find the entry at 4d0:227
> The semaphore name is SEM\NET\WKSTA\RELOGON.SEM
```

```
--------------------------------------------
```

---

**The MUX RAMSEM**

In a MUX the RAMSEM id is recorded as a hob:offset.

In this example we look at the RAMSEM being waited on by slot 1f.

```
> The MUX table entry:
> slot = 1f, type = 2, hob= 03fa, offset=0758
```

```
0400:000048d9 e2 48 1f 00 02 58 07 fa 03 bH...X.z.
# db muxtable+(9*4) l9
```

```
>> Use .MOC to find the linear address
```

```
# .moc 3fa
*har     par      cpg        va    flg next prev link hash hob   hal
039b %fef23f5c 00000010 %1a350000 379 0413 039c 0000 0000 03fa 0000  hco=00c45
hob   har hobnxt flgs own  hmte  sown,cnt lt st xf
03fa  039b 0000  082c 03fb 03fb  0000 00  00 00 00 shared    e:pmshapi.dll
hco=0c45 pco=ffe77d74  hconext=00e0 hptda=0873 f=16 pid=00e0 e:cmd.exe
```

```
>> This is owned by PMSHAPI. LN gives a label.
```

```
# ln %1a350750
%1a350750 PMSHAPI ASEMRS + 8
```

---
MUX Physical RAMSEM

In this example the MUX wait entry is for a physical address of a RAMSEM. A physical address would be used where the RAMSEM is in instance data - that makes it unique among RAMSEMs providing the RAMSEM is not swappable.

In this example the waiting slot is 1b

```
>> Mux table entry for slot 1b, type=3 (physical RAMSEM)
>> The physical address of the RAMSEM is %00f1a6da
>> We need to determine the owner of this address.
```

```
0400:0000493c be 48 1b 00 03 da a6 f1-00
```

```
>> Display the page frame structure for frame 00f1a:
```

```
# .mp fla
ffe24538 InUse: pVP=ff4076ce RefCnt=0003 Flg=0 ll=01 sl=00 Blk=0006a Frame=00f1a
```

```
>> Now display the virtual page structure to see who has backed this frame:
```

```
# .mv %ff4076ce
VPI=057b pVP=ff4076ce SOW Frame=0f1a Flg=9d0 HobPg=0000 Hob=03df Ref=011
```

```
>> RAMSEM is at %1a260000+0000000+6da
>> RAMSEM is owned by pmwin.dll
```

```
>> .moc 3df
```

```
*har     par      cpg        va    flg next prev link hash hob   hal
0382 %fef23d36 00000010 %1a260000 379 0381 0383 0000 00 03df 0000  hco=00f37
03df 0382 0000 082c 03da 03da 0000 01 00 00 shared e:pmwin.dll
```

```
>> RAMSEM is at %1a260000+0000000+6da
>> RAMSEM is owned by pmwin.dll
```

```
# ln %la2606da
No Symbols Found
```

```
>> LN doesn't work so thunk to a selector:offset and try again
```

```
>> cheat by looking up the selector assigned to pmwin in its segment table:
```

```
# .lmo 3da
```

```
obj vsze vbase flags ipagemap cpagemap hob sel
0001 000066f8 1a1b0000 80005025 00000001 00000010 03e9 d0df r-x shr alias conf
0002 0000c24e 1a1c0000 80005025 00000001 0000000d 03e8 d0e7 r-x shr alias conf
0003 00008eb4 1a1d0000 80005025 0000000e 00000009 03e7 d0e6 r-x shr alias conf
0004 0000b6e2 1a1e0000 80005025 0000000f 0000000c 03e6 d0f7 r-x shr alias conf
0005 0000eb10 1a1f0000 80005025 00000033 0000000f 03e5 d0ff r-x shr alias conf
0006 00006292 1a200000 8000d025 00000042 00000007 03e4 d106 r-x shr alias conf
0007 00003738 1a210000 8000d025 00000049 00000004 03e3 d10e r-x shr alias conf
0008 000010c5 1a220000 80009025 0000004d 00000002 03e2 d116 r-x shr alias iopl
0009 000124d4 1a230000 80003025 0000004f 00000013 03e1 d11f r-x shr alias big
000a 000070ca 1a250000 80001025 00000062 00000008 03e0 d12f r-x shr alias big
000b 000000da 1a260000 80001063 0000006a 00000001 03df d137 r-x shr prel alias
000c 00000178 1a270000 80001063 0000006b 00000002 03de d13f rw- shr prel alias big
000d 0000023f 1a280000 80001063 0000006d 00000003 03dd d147 rw- shr prel alias big
000e 00000644 1a290000 80001063 00000070 00000002 03dc d14f rw- shr prel alias big
000f 00000142 1a2a0000 80001063 00000072 00000001 03db d157 rw- shr prel alias big
0010 00000018 1a2b0000 80002063 00000073 00000001 03d9 d15f rw- shr prel big
0011 0000038b 16100000 80002079 00000074 00000001 051e b087 r--- rsr disc shr prel big
0012 00000dcc 161c0000 80002069 00000075 00000001 0509 b0e7 r--- rsr disc shr prel big
0013 0000ffbc 16210000 80002029 00000076 00000010 0504 b10f r--- rsr disc shr big
0014 000002f0 00000020 80002039 00000086 00000001 0000 0000 r--- rsr disc shr big
0015 00003524 16120000 80002029 00000087 00000004 051b b097 r--- rsr shr big
```

```
# ln d137:6dad137:00006da PMWIN MSGQUEUESEM1
```
Structured Semaphores

We have discussed the following type of semaphore:

- RAMSEM
- SYSSEM
- FSRAMSEM
- PMFSRAMSEM

There are three others that occur with regularity in the system:

- KSEM
- 32-bit SEM
- GRESEM/PMSEM

The Kernel Semaphore (KSEM)

The kernel semaphore is a RAMSEM and EVENT SEM with accountability in-built.

Many system control block have imbedded KSEMs. Included among these are the PTDA and MFT.

Some KSEMs are allocated out of the kernel heaps and have the owner mnemonic KSEM.

When a thread blocks on a KSEM the address (or handle) of the KSEM, or a field within the KSEM, is used as the BlockId depending upon the KSEM type.

- For MUTEX KSEMs: The BlockId is the address of the beginning of the KSEM structure.
- For Shared KSEMs: The BlockId is the address of the Pending Readers count field within the KSEM structure.
- For Exclusive KSEMs: The BlockId is the address of the Pending Writers count field within the KSEM structure.

Under the debug (ALLSTRICT) kernel the KSEM contains an additional signature 'KSEM'. Always check a BlockId address to see if the 'KSEM' signature is present.

.D KSEM will format the KSEM.

In this example we look at Slot 6c to find out why it will not run.

```
# .pb 6c
Slot Sta BlockID Name   Type Addr Symbol
006c blk 7bdfc910 DEMO1

# .m 7bdfc910
har  par  cpg va flg next prev link hash hob hal
0087 %fef1fba4 00000082 %7bdf5000 121 0085 0088 0000 0000 0089 0000 =0000
hob har hobnxt flgs own hmte sown,cnt lt st xf
0089 0087 0000 0325 ffeb 0000 0000 00 00 00 00 ptda

>> This thread is blocked on an address in (its) PTDA. All PTDA
>> semaphores are KSEMs.
```

```
# .d KSEM %7bdfc910
Signature : KSEM Nest: 0001
Type : MUTEX
Flags : 00
Owner : 0041 PendingWriters: 0001
```
So the owner is Slot 41. Lets look at him to see what he's up to.

```
Slot  Sta BlockID  Name     Type        Addr        Symbol
0041  blk 04085ca7 DEMO1
```

Slot 41 is blocked waiting for some file system activity to complete. We looked at this slot some time ago and found out that it was waiting to close a device driver.

The 32-Bit Semaphore Event and Mutex Semaphores

Block ids for 32-bit sems point to kernel heap allocated structure with object mnemonic `semstruc`.

`.PB` under the KDB usually identifies these as `SEM32`, but DF doesn't.

`.D SEM32` will format a 32-bit semaphore structure.

There are several structures that relate to 32-bit semaphores. Each of these is allocated from the kernel heaps and is assigned the following meaningful owner id mnemonics:

- `semmuxq (0xffbe)`: Semaphore Mux Queue. This records instances of single event or mutex semaphores being also waited on in a mux wait.
- `semopenq (0xffbf)`: Semaphore Open Queue. This tracks all processes that have opened a 32-bit semaphore.
- `semrec (0xffc0)`: SemRecord. This is a system copy of the user's SemRecord structure, which was created when a Mux wait was declared. It correlates user semaphore ids with semaphore handles.
- `semstr (0xffc1)`: The semaphore name string.
- `semstruc (0xffc2)`: The main 32-bit structure. The address of this forms the BlockId when a thread waits on a 32-bit semaphore.

Of the associated structures listed above the Open Queue and Mux Queue may be formatted using:

```
.D OPENQ
.D MUXQ
```

In this example we look at the BlockId slot 42 is waiting on.

```
Slot  Sta BlockID  Name     Type        Addr        Symbol
0042  blk fe0bf91c PMSHL32
```

> check owner of blockid

```
*har par cpg va flg next prev link hash hob hal
0003 %fef1fa70 000000010 %7bf27000 129 0078 0077 0000 0000 007b 0000  sel=0408
hob har hobnxt flgs own hmte sown,cntl lt st xf
007b 0079 0000 0324 ffa1 0000 0000 00 00 00 00 sft
```

>> kernel swappable heap. Check current user of heap block.

```
dd %fe0bf91c-10
%fe0bf90c ffbf000c 00010008 fe0bff20 ffc20014
%fe0bf91c 00000011 00000000 fef1ef94 fcae5a28
```
This is a 32-bit Semaphore

Type: Shared Event
Flags: Reset
Post Count: 0
pOpenQ: fef1ef94
pName: fcae5a28
Create Addr: ffbf0010

For interest look for the owner of the OPENQ:

For interest look for the owner of the pName:

PMSEM/GRESEM
32-bit PM (WARP) and Graphics Engine use a composite semaphore structure to serialise their resources.

This semaphore has the structure:

+0x0  7 byte Signature. 'PMSEM' for PMWIN and 'GRESEM' for PMSGRE
+0x7  386 semaphore byte (PM uses the bts instruction on this under 386 processors otherwise it uses the 486 cmpxchg on the pid/tid)
+0x8  Owner pid (word)
+0xa  Owner tid (word)
+0xc  Owner nested use count (long)
+0x10 Number of waiters
+0x14 Number of times sem used (zero unless Debug version of PM)
+0x18 Handle for event semaphore
+0x1c Address of caller (zero unless Debug version of PM)

PM uses a technique of polling this semaphore by waiting on the imbedded event semaphore handle for a limited time.

This technique has the advantage of speed combined with accountability but a thread waiting for a PMSEM or GRESEM may appear blocked, ready or running depending on the polling cycle. However it will be executing in a routine with a name such as PMRequestMutexSem. If the PMMERGE symbols are loaded this is readily detected.

The PM and GRE SEMs are contiguous and located at label pmSemaphores.

The handle (linear address) of the PM/GRE Semaphore is passed on entry to PMRequestMutexSem and tends to be retained in the EDX register.

The following semaphores are defined by PM:

0  PMSEM ATOM
1  PMSEM USER
2  PMSEM VISLOCK
3  PMSEM DEBUG
4  PMSEM HOOK
5  PMSEM HEAP
6  PMSEM DLL
7  PMSEM THUNK
8  PMSEM XLCE
9  PMSEM UPDATE
10 PMSEM CLIP
11 PMSEM INPUT
12 PMSEM DESKTOP
13 PMSEM HANDLE
14 PMSEM ALARM
15 PMSEM STRRES
16 PMSEM TIMER
17 PMSEM CONTROLS
In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
 003a 000d 0005 000d 000a rdy 0200 abd61000 abe4b5b4 abe2ee60 0ee4 11 PMSHL32
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .p 3a
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
 003a 000d 0005 000d 000a rdy 0200 abd61000 abe4b5b4 abe2ee60 0ee4 11 PMSHL32
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .r
eax=13e30025 ebx=00000000 ecx=000a000d edx=13e7b4d4 esi=ffffffff edi=0068e55c
eip=1bd0d7ea esp=00637f44 ebp=00637f60 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001ad000
tid/pid 10
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .p 42
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
 042  0010 0005 0010 0001 blk 0500 abd69000 abe4c19c abe2fe20 0ed8 13 MRFILEPM
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .pb 42
Slot  Sta BlockID  Name     Type        Addr        Symbol
 042  blk fdf8841c MRFILEPM
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .m %fdf8841c
har     par      cpg        va    flg next prev link hash hob   hal
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .p 3a
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
 003a 000d 0005 000d 000a rdy 0200 abd61000 abe4b5b4 abe2ee60 0ee4 11 PMSHL32
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .r
eax=13e30025 ebx=00000000 ecx=000a000d edx=13e7b4d4 esi=ffffffff edi=0068e55c
eip=1bd0d7ea esp=00637f44 ebp=00637f60 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001ad000
tid/pid 10
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .p 42
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
 042  0010 0005 0010 0001 blk 0500 abd69000 abe4c19c abe2fe20 0ed8 13 MRFILEPM
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .pb 42
Slot  Sta BlockID  Name     Type        Addr        Symbol
 042  blk fdf8841c MRFILEPM
```

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```plaintext
# .m %fdf8841c
har     par      cpg        va    flg next prev link hash hob   hal
```
Involuntary Suspension

In this section we discuss the mechanisms involved when a thread involuntarily gives up CPU processing time. That is, another thread independently causes a thread not to receive or to give up its time-slice.

The mechanisms available that cause suspension are:

**Pre-emption**

Another thread of a high priority becomes ready.

The suspended thread becomes ready and the pre-empting thread runs.

**Note:**

Pre-emption is not possible when running in kernel-mode (specifically when InDos is non-zero, which is set shortly after entry to the kernel). Within the kernel co-operative multi-tasking operates: threads must yield explicitly (call the scheduler) to give up the processor. This applies equally to device drivers and file system drivers, which also run in kernel mode. Physical Device Drivers may use DevHip_Yield and DevHip_TCYield to give up the processor to other threads.

**Critical Section**

Another thread in the same process enters critical section.

The critical section thread runs and none of the other threads will run except if a signal 'fires'. If another ready thread in the same process is selected by the dispatcher for running it is held on a temporary queue with its status set to **crt**.

**Note:**

The Critical Section thread has **run** status.

**DosSuspendThread**

Another thread in the same process has issued DosSuspendThread.

The suspending thread runs and the suspended thread enters **frz** state.
Freeze Process

Either a Session Manager switch is in progress, a new process has been created suspended, a Virtual Device Driver has called the VDHFreezeVDM helper routine or the DosDebug DBG_C_Freeze command has been executed against a debuggee process.

The frozen process has a state of frz in all its threads.

Voluntary suspension is indicated by the blk state.

When a thread is suspended involuntarily it will normally be in one of the following states:

- **rdy**: Ready and waiting to run.
- **crt**: Ready but prohibited from dispatch by a critical section thread.
- **frz**: Frozen or Suspended by freeze-process or DosSuspendThread.

The remaining six thread states related to transient system processing on behalf of a thread. These are:

- **dly**: Delayed wake-up. Multiple threads have been woken from a blocked state because they were all waiting on the same BlockId and a multiple wake-up was specified to ProcRun. Each delayed thread is queued pending scheduling where priority recalculation and the thread's ring 0 stack is checked for presence in memory. If all is well then the thread is placed on the ready queue pending dispatch. If not, then the thread is placed on the TSD Daemon's queue for paging in the thread's TSD (ring 0 stack).

- **tsd**: The thread is on the TSD Daemon's queue waiting for ring 0 stack page-in. The Daemon runs as an internal thread, which is labelled *tsd by the .P command. This thread is responsible for calling the page manager to page in a thread's TSD. Because a paging operation involves I/O and is therefore relatively slow, this operation is performed under the control of a separate thread. This allows other threads to be processed while the paging operation takes place.

- **gsk**: Get Stack request in progress. The TSD Daemon is waiting for the Page Manager to signal completion of the paging I/O operation. Effectively a thread in this state is blocked waiting for completion of a TSD paging I/O request.

- **bst**: Boosted Ready State. When the TSD page-in completes successfully, the thread is placed on the dispatcher's ready queue with a priority boost. This condition is indicated by the boosted ready state. Strictly speaking this is not an independent state since no operation is required to take the thread from bst to rdy.

- **bad**: TSD page-in request has failed. This is a serious and terminal condition, which is not expected to occur. It is possible that an I/O error has occurred during the TSD page-in request.

---

The null state occurs very fleetingly during thread creation and termination. It signifies that the thread's environment is incomplete.

The complete set of scheduler states for a finite state machine, which is illustrated in the following diagram.
Pre-emption and Priority Calculation

A thread is pre-empted when higher priority work becomes ready to process. Under normal circumstances the pre-empting thread will run then give up its time-slice and eventually the original thread will be re-scheduled.

It is possible for a thread not to be re-scheduled if a higher priority thread will not give up the processor. However, the OS/2 scheduler applies dynamic boost to priorities according to resource requirements and makes priority comparisons based on a calculated priority. The elements involved in the priority calculation are the following:

**TCBPriClass**

The thread's priority class. There are four classes, which in order of priority are:

3  Time-critical
4  Foreground Server (or fixed high)
2  Regular
1  Idle

**TCBPriLevel**

The priority delta which may range from 0x00 to 0x1f.

**TCBPriClassMod**

The priority boosts which may be any of the combined values:
0x04  Keyboard Boost
0x08  CPU Starvation Boost
0x10  Device I/O Boost
0x20  Foreground Boost
0x40  Window Boost
0x80  VDM Simulated Interrupt.

TCBPriClass

The minimum allowed priority. Normally 0 but set when priority inversion becomes a possibility. This is discussed later.

Priority is calculated by forming an index by ORing TCBPriClass and TCBPriClassMod and reading a constant value from the priority table. The low byte of this is then further ORed with the TCBPriLevel.

The following diagram shows the priority table.

Table Index = (TCBPriClass | TCBPriClassMod)

Starved         08 --------------------------------------------------+
Device I/O      10 -------------------------------------------------|--
Foreground      20 ------------------------------------------------|--|--
Window          40 -----------------------------------------------|--|--|--
VDM Interrupt   80 ----------------------------------------------|--|--|--|--

+++ TCBPriClass

<table>
<thead>
<tr>
<th>Not Keyboard</th>
<th>Keyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Idle</td>
<td>Server Idle</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x300, 0x100, 0x200, 0x800, 0x300, 0x200, 0x800, // -----</td>
<td></td>
</tr>
<tr>
<td>0x62f, 0x100, 0x61f, 0x800, 0x62f, 0x100, 0x61f, 0x800, // ----S</td>
<td></td>
</tr>
<tr>
<td>0x72f, 0x100, 0x71f, 0x800, 0x72f, 0x100, 0x71f, 0x800, // ---D-</td>
<td></td>
</tr>
<tr>
<td>0x72f, 0x100, 0x71f, 0x800, 0x72f, 0x100, 0x71f, 0x800, // ---DS</td>
<td></td>
</tr>
<tr>
<td>0x300, 0x100, 0x300, 0x800, 0x300, 0x100, 0x400, 0x800, // ---F--</td>
<td></td>
</tr>
<tr>
<td>0x62f, 0x100, 0x61f, 0x800, 0x62f, 0x100, 0x61f, 0x800, // ---F-S</td>
<td></td>
</tr>
<tr>
<td>0x74f, 0x100, 0x73f, 0x800, 0x74f, 0x100, 0x73f, 0x800, // --FD-</td>
<td></td>
</tr>
<tr>
<td>0x74f, 0x100, 0x73f, 0x800, 0x74f, 0x100, 0x73f, 0x800, // --FDS</td>
<td></td>
</tr>
<tr>
<td>0x500, 0x100, 0x500, 0x800, 0x500, 0x100, 0x500, 0x800, // --W---</td>
<td></td>
</tr>
<tr>
<td>0x62f, 0x100, 0x61f, 0x800, 0x62f, 0x100, 0x61f, 0x800, // --WF--</td>
<td></td>
</tr>
<tr>
<td>0x74f, 0x100, 0x73f, 0x800, 0x74f, 0x100, 0x73f, 0x800, // --WF-S</td>
<td></td>
</tr>
<tr>
<td>0x74f, 0x100, 0x73f, 0x800, 0x74f, 0x100, 0x73f, 0x800, // --WFD-</td>
<td></td>
</tr>
<tr>
<td>0x74f, 0x100, 0x73f, 0x800, 0x74f, 0x100, 0x73f, 0x800, // --WFDS</td>
<td></td>
</tr>
</tbody>
</table>

Notes

VDM Simulated interrupts always result in a value of 0x800

Foreground server class is not affected by the keyboard boost.

Time-critical class is not affected by any boosts.

Idle class is not affected by any boosts.

By examining the priority table it is clear that idle class will always be pre-empted by any other class.

Time-critical class can never be pre-empted by any other class.

Time-critical threads can only be pre-empted by other time-critical threads with a higher delta.

Server and Regular class threads may pre-empt each other depending on priority boosts and delta.
The key to looking at pre-emption problems is to look for other CPU bound threads of a higher priority. In particular time-critical threads.

.P displays the current calculated priority for each thread.

Critical Sections

When a thread enters critical section it effectively suspends all other threads in its process. There is an exception to this. If a signal is sent to the process and a signal handler is registered, then thread 1 will be dispatched to run the signal handler regardless of critical section.

The critical section thread may voluntarily block. Other threads may attempt to become ready. If this happens the dispatcher will temporarily suspend them in *crt state. The appearance of the *crt state certainly guarantees that another thread in the same process is in critical section. However, the converse in not true: the absence of *crt does not preclude another thread from being in a critical section.

If a thread running in Critical Section blocks on a resource owned by any other thread in the same process then a deadlock will result. Because of this it is unwise to call any System API when in Critical Section.

Thread running in Critical Section have their TCB address stored in their process's PTDA at ptda_pTDBCritSec.

The following example illustrates locating the critical section thread in a process.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Pid</th>
<th>Ppid</th>
<th>Csid</th>
<th>Ord</th>
<th>Sta</th>
<th>Pri</th>
<th>pTSD</th>
<th>pPTDA</th>
<th>pTCB</th>
<th>Disp</th>
<th>SG</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0001</td>
<td>blk</td>
<td>0100</td>
<td>ffe4b000</td>
<td>ffe4c7dc</td>
<td>7b49c020</td>
<td>0e44</td>
<td>*tsd</td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0002</td>
<td>blk</td>
<td>0200</td>
<td>7a49e000</td>
<td>ffe4c7d7</td>
<td>7b49c1d8</td>
<td>0f54</td>
<td>*ctxh</td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0003</td>
<td>blk</td>
<td>0200</td>
<td>7a49f000</td>
<td>ffe4c7dc</td>
<td>7b49c390</td>
<td>0f24</td>
<td>*kdb</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0004</td>
<td>blk</td>
<td>0200</td>
<td>7a4a0000</td>
<td>ffe4c7dc</td>
<td>7b49c548</td>
<td>0f40</td>
<td>*lazyw</td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0005</td>
<td>blk</td>
<td>0200</td>
<td>7a4a1000</td>
<td>ffe4c7dc</td>
<td>7b49c678</td>
<td>0f04</td>
<td>*LAMSSEX</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0006</td>
<td>blk</td>
<td>0200</td>
<td>7a4a2000</td>
<td>ffe4c7dc</td>
<td>7b49c8b8</td>
<td>0ebc</td>
<td>*ager</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0007</td>
<td>blk</td>
<td>0200</td>
<td>7a4a3000</td>
<td>ffe4c7dc</td>
<td>7b49cf98</td>
<td>01</td>
<td>PMSHL32</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0008</td>
<td>blk</td>
<td>0200</td>
<td>7a4a4000</td>
<td>ffe4c7dc</td>
<td>7b49f040</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0009</td>
<td>blk</td>
<td>0200</td>
<td>7a4a5000</td>
<td>ffe4c7dc</td>
<td>7b49f1f8</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0010</td>
<td>blk</td>
<td>0200</td>
<td>7a4a6000</td>
<td>ffe4c7dc</td>
<td>7b49f3b0</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0011</td>
<td>blk</td>
<td>0200</td>
<td>7a4a7000</td>
<td>ffe4c7dc</td>
<td>7b49f568</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0012</td>
<td>blk</td>
<td>0200</td>
<td>7a4a8000</td>
<td>ffe4c7dc</td>
<td>7b49f720</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0013</td>
<td>blk</td>
<td>0200</td>
<td>7a4a9000</td>
<td>ffe4c7dc</td>
<td>7b49f7bf</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0014</td>
<td>blk</td>
<td>0200</td>
<td>7a4a0000</td>
<td>ffe4c7dc</td>
<td>7b49f8d8</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0015</td>
<td>blk</td>
<td>0200</td>
<td>7a4a1000</td>
<td>ffe4c7dc</td>
<td>7b49fa90</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0016</td>
<td>blk</td>
<td>0200</td>
<td>7a4a2000</td>
<td>ffe4c7dc</td>
<td>7b49fb48</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0017</td>
<td>blk</td>
<td>0200</td>
<td>7a4a3000</td>
<td>ffe4c7dc</td>
<td>7b49fc48</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0018</td>
<td>blk</td>
<td>0200</td>
<td>7a4a4000</td>
<td>ffe4c7dc</td>
<td>7b49fd10</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0019</td>
<td>blk</td>
<td>0200</td>
<td>7a4a5000</td>
<td>ffe4c7dc</td>
<td>7b49ff88</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0020</td>
<td>blk</td>
<td>0200</td>
<td>7a4a6000</td>
<td>ffe4c7dc</td>
<td>7b4a0170</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0021</td>
<td>blk</td>
<td>0200</td>
<td>7a4a7000</td>
<td>ffe4c7dc</td>
<td>7b4a0328</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0022</td>
<td>blk</td>
<td>0200</td>
<td>7a4a8000</td>
<td>ffe4c7dc</td>
<td>7b4a04e0</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0023</td>
<td>blk</td>
<td>0200</td>
<td>7a4a9000</td>
<td>ffe4c7dc</td>
<td>7b4a0678</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0024</td>
<td>blk</td>
<td>0200</td>
<td>7a4a0000</td>
<td>ffe4c7dc</td>
<td>7b4a0850</td>
<td>01</td>
<td>CMD</td>
<td></td>
</tr>
</tbody>
</table>
In this example pid 15 is stuck, threads are either blocked or suspended by critical section.

The critical section thread has blocked on a RAMSEM whose address is \texttt{00bf:0024}. Since the selector is less than \texttt{2007} this has to be in its private arena. This is significant: only another thread in the same process could possibly post this semaphore.

Suspension and Freezing

Suspension is achieved by any thread in a process calling \texttt{DosSuspendThread}. There is no accounting information associated with this API. One must examine all threads in the process to see if they are functioning correctly.

Freezing occurs for a number of reasons:

- A new process has been created with the thread initially suspended.

- The Session Manager (Shell Process 1) has used \texttt{DosSystemService} to freeze all threads of a process while a screen group switch occurs.

- A Virtual Device Driver has called \texttt{VDHFreezeVDM}.

- A Debug thread has called \texttt{DosDebug} using the \texttt{DBG_C_Freeze} command.

Again, there is no accounting information kept for this state.

If a single thread exists in the frozen process, check the parent process's threads for correct functioning.

If all threads are frozen check the Shell Process 1 for correct processing.

Priority Inversion

Priority Inversion is a hybrid situation that involves both the involuntary and voluntary suspension of two threads.

Consider the following:
A high priority thread is blocked on a resource.

A low priority thread owns the resource on which the high priority thread is blocked.

An independent thread of intermediate priority is running.

Thread 1
High
blk

waits
on
SEM
owned < - - - - - - - Medium
by thd 2
V

Thread 2
Low
rdy

Thread 1 will not run until thread 2 gets a time-slice that allows it to run and release the semaphore thread 1 is waiting for.

Since thread 3 is a higher priority than thread 2 and is CPU bound, thread 2 never runs, nor does thread 1.

Thread 1’s priority has effectively been reduced to that of thread 2’s by a lower priority thread - thread 3. Thread 1 is said to have its priority inverted with respect to thread 3.

The Kernel implements an automatic inversion protection mechanism whenever a process blocks using a KSEM. Essentially this amounts to boosting the KSEM owner's thread priority by setting TCBPriorityMin to be just greater than the waiter's priority. This mechanism is implemented by the following three routines:

TKEnterInversion
Called to protect against priority inversion. For example, when a mutex KSEM is obtained increments TCBcBoostLock.

TKExitInversion
When an inversion protected KSEM is released TCBcboostlock is decremented. When TCBcBoostlock is zero and TCBPriorityMin and is not zero then it is set to zero and the priority recalculated.

TKDeclareInversion
Used to set the minimum priority of a thread to be waited on. If owner's TCBPriorityMin < waiter's TCBPriority and owner is in ready state then the owner's TCBPriorityMin to the waiter's TCBPriority+1.

For this mechanism to work, it must be possible to determine ownership from the semaphore so that TKDeclareInversion can determine which thread’s priority to alter. It is also necessary to be able to determine whether raising the priority of thread will lead to other synchronisation problems or deadlocks through race conditions. Since the Kernel is a special case, and because pre-emption cannot occur while running in kernel mode, the kernel limits inversion protection to the KSEM only. Outside kernel mode, inversion is automatically protected against (for regular and foreground server threads) by application of the starvation priority boost.

-------------------------------

Program Design Issues and Weaknesses

The following hit-list identifies potential weaknesses in program design that can lead to hang symptoms or serialisation problems:
1. Manipulation of thread priorities for the purpose of serialisation or sequencing execution is haphazard at best. At worst the performance of the entire system can be jeopardised.

The following guidelines should be applied when considering priority manipulation:

- Use priority delta to tell the system the relative importance of an application's threads.
- Avoid priority class manipulation. Priority class tends to specify the relative importance of a thread with respect to all other threads in the system.
- Avoid the use of time-critical priority. By setting this class, a thread is assuming the position of utmost importance in the system. This may not be a valid assumption for some system configurations and some users.
- If priority class manipulation is desirable under some circumstances, then it should be parameterised so that it can be controlled as an option by the user.

2. If a window of exposure exists it will be exposed.

3. Any common resource that is ever modified must have an associated lock or serialisation mechanism.

4. Locks (serialisation techniques such as semaphores) that are concurrently held and waited on must be obtained in an established order.

5. Simplistic approach (one lock) forces work to be channelled through a single-queue. Therefore design locks at the lowest level of contention.

6. Distinguish process/data/repository serialisation otherwise an inconsistent system of locks may result:

   - Process locks are required where a only single instance of a process is allowed to operate. For example:
     - Finite State Machine state transitions;
     - Some FSM state users;
     - Any non-reentrant process.

   - Distinguishing Repository locks allows the repository is updated:
     - disk/directory reorganisation while file data is in use.
     - physical page assignments are allowed to change while data is in use - swapping

7. Data optimisation: Artificial association of unrelated data items imposes serialisation constraints that will have two possible effects:

   - This necessitates unrelated processes to serialise.
   - Serialisation may lead to unavoidable deadlocks.

8. Code optimisation: imposes process lock constraints in a similar way that data optimisation does.

9. O-O tends to hide the data repository and structure. May even hide the process. Therefore designers need to consider whether locks are managed internally, within the object or explicitly. It may not be possible to handle the locks internally, because the context in which an instance method is being use (that is, the process) is not discernible from within the object.

Worked Examples

The following collection of worked examples illustrate how to use the debugging tools, in particular the Dump Formatter and Kernel Debugger to obtain information from a system under diagnosis.

The following topics are included:

Finding File System information
This gives techniques for obtaining open file information and correlating open file names to handles and vice versa and finding out about record locking.
How to Find File System Information

This section gives a basic overview of the file system control blocks, and shows how to answer the following questions:

1. What file system objects are open in a given process?
2. What file system objects are open in a VDM?
3. What file corresponds to a given handle?
4. What processes have opened a given file system object?

If the reader is unfamiliar with this subject then the sections that follow should be read in order:

These topics now follow:

Finding files from handles
Finding files from handles in a VDM
Finding handles from file names
The Record Lock Record

Note:

The examples included in this section are worked on an OS/2 2.11 system. For OS/2 WARP the same techniques work, however the SFTs, whilst they may be located from the SAS in the same manner, they are allocated in segments that are mapped by different selectors. The effect of this is that short-cut techniques used to locate WARP SFTs may need to be re-worked.

Finding Files From Handles

Open file system objects (files, named pipes, devices etc.) are represented by the SFT control block. The SFT contains three sections:

- Kernel data
- File system independent data
- File system dependent data

The kernel data section contains information to link the SFT to other system control blocks and to make the SFT usable by Kernel APIs. Of principle interest in this section are flags, handle, and pointer to the MFT and a chain pointer to other SFTs that represent other open instances of the same object. The kernel data is split into two discontiguous sections at each end of the SFT.

The file system independent data section contains information common to all FSDs needed to drive the file system. Of principle interest are the file attributes, open mode flags, opening process id and handle to the associated VPB.
The file system dependent data section is, as the name suggests, a work area private to the FSD that manages the file system object.

Note:

The .D SFT command formats the SFT always as if it is a FAT file. The information displayed in the file system dependent section may be misleading for non-FAT objects. The names of the fields formatted by .D SFT are prefixed sfdFAT_ for the file system dependent data so make it clear which information to treat with circumspection. The kernel and file system independent data name are prefixed with sf_ and sfi_ respectively.

When a file system object is opened, DosOpen returns a handle that represents the open object for all subsequent file system manipulation by the process until the object is closed. This handle is unique only within process and is referred to as the JFN. In protect mode processes the JFN is a 16-bit entity. In VDMs, however, to be consistent with DOS the JFN is an 8-bit entity, which may be correlated to the real JFN through a table in the PDB. This is illustrated later.

Each open file system object is also known by a system-wide unique handle, the SFN. Once the SFN is known then the corresponding SFT may be located and thence all file system information relating to the object.

Each process is assigned by default a table of 20 words, which is indexed by the JFN. Each word of the JFN_table contains the corresponding SFN for the open file. The default JFN_table is imbedded within the PTDA. Prefixing the JFN_table is a double-word pointer (JFN_ptable) that points to this table. If the table is expanded (using DosSetMaxFH) then JFN_ptable is updated to point to the current JFN_table.

The key to finding information about open object in a given process is to locate JFN_table and JFN_table. Since both of these fields are part of the PTDA they may be referred to by name as symbols for the current (system) context only. For other contexts we may still use the PTDA symbols but in a relative fashion. The PTDA symbols are defined for the current process, which means that to use them successfully for another process, one must relocate them to the PTDA one wishes to reference. This is easily done by subtracting the label PTDA_START from the desired symbol, then adding the address of the PTDA one wishes to see. For example: to see the jfn_table field, enter:

dw <ptda address>+jfn_table-ptda_start L2.

The relationships between the JFN_table, PTDA and the SFT is illustrated in the following diagram:
File System Control Block Relationships

In the examples that follow, we explore the relationships between each of the major file-system control blocks. These relationships are illustrated in the following diagrams.

Open File
Open Device
Open Device - System View
### Finding Files From Handles - Example

In the following example we choose to discover all the open file system objects in process 19, which happens to be running the IPFC compiler.

```bash
>>> List all the thread slots in the system to find IPFC
# .p
Slot Pid Ppid Csid Ord STA Pri pTSD pPTDA pTCB Disp SG Name
0001 0001 0000 0000 0001 blk 0100 ff3a000 ff3e7d4 ff3c61c 1e7c 00 *ager
0012 0001 0000 0000 0013 blk 0200 7b7e000 ff3e7d4 7b9a8020 1f3c 00 *ager
0003 0000 0000 0003 0003 blk 0200 7b7bac0 0000 ff3e7d4 7b9a84f0 1f3c 00 *ctkhs
0004 0001 0000 0004 0004 blk 081f 7b7ae000 ff3e7d4 7b9a8390 1f48 00 *kdb
0005 0001 0000 0005 0005 blk 0800 7b7b0000 ff3e7d4 7b9a8548 1f20 00 *lazyw
0006 0001 0000 0006 0006 blk 0800 7b7b2000 ff3e7d4 7b9a8750 1f3c 00 *asyncr
0009 0002 0000 0002 001 blk 021f 7b7b8000 7b9c4020 7b9a8c28 00 LOGDAEM
0008 0003 0001 0003 0001 rdr 061f 7b7b6000 7b9c484c 7b9a87a0 0eb8 01 PMSHL32
0010 0003 0001 0003 0002 blk 0800 7b7bc000 7b9c484c 7b9a89f0 01 PMSHL32
0010 0003 0001 0003 0003 blk 0800 7b7be000 7b9c484c 7b9a8950 01 PMSHL32
0004 0001 0003 0004 blk 0800 7b7c0000 7b9c484c 7b9a9300 01 PMSHL32
0005 0001 0003 0005 blk 0800 7b7c2000 7b9c484c 7b9a94c0 01 PMSHL32
0017 0003 0001 0006 blk 0200 7b7d0000 7b9c484c 7b9a88b8 14ec 01 PMSHL32
0011 0003 0001 0007 blk 0200 7b7d8000 7b9c484c 7b9a99e8 14ec 01 PMSHL32
0012 0003 0001 0008 blk 0200 7b7e0000 7b9c484c 7b9a9ba0 01 PMSHL32
0013 0003 0001 0009 blk 0200 7b7e2000 7b9c484c 7b9a9d58 01 PMSHL32
0014 0003 0001 0010 blk 0200 7b7e4000 7b9c484c 7b9a9f10 01 PMSHL32
0015 0003 0001 0011 blk 0200 7b7e6000 7b9c484c 7b9aa0c8 01 PMSHL32
0016 0003 0001 0012 blk 0200 7b7e8000 7b9c484c 7b9aa280 01 PMSHL32
0017 0003 0001 0013 blk 0200 7b7d0000 7b9c484c 7b9a9a38 1eab 01 PMSHL32
0018 0003 0001 0014 blk 0200 7b7d6000 7b9c484c 7b9a95f0 01 PMSHL32
0019 0003 0001 0015 blk 0200 7b7d8000 7b9c484c 7b9a97a8 01 PMSHL32
0010 0003 0001 0016 blk 0200 7b7d0000 7b9c484c 7b9aa960 0bac 01 PMSHL32
```

---

In the following example we choose to discover all the open file system objects in process 19, which happens to be running the IPFC compiler.
>>> slot so we have to refer to PTDA symbols relative to pPTDA
>>> First establish whether JFN_table has been expanded?
# dw %7b9ca230 + jfn_ptable-ptda_start l2
%7b9ca9e 8d8a 0030
>>> No it hasn't - it's still based on selector 30 and therefore
>>> still imbedded in the PTDA at label JFN_table.
>>> Note: we can't display it as 30:fd8a since selector 30
>>> aliases the current system PTDA, hence:
# dw %7b9ca230 + jfn_table-ptda_start l14
%7b9ca7e6 0027 0027 0027 0074 002a 0072 0077 0068
%7b9ca7f6 0015 0041 0069 007f ffff ffff ffff ffff
%7b9ca806 ffff ffff ffff ffff
>>> These are the SFNs that correspond to JFNs 0000 through 0014.
>>> In fact the highest JFN currently open in this process is 000b
>>> which corresponds to SFN 007f

>>> Next we locate the STF. From the SAS we look for the SFT selector:

# .a
--- SAS Base Section ---
  SAS signature: SAS
  offset to tables section: 0016
  FLAT selector for kernel data: 0168
  offset to configuration section: 001E
  offset to device driver section: 0020
  offset to Virtual Memory section: 002C
  offset to Tasking section: 005C
  offset to File System section: 0074
  offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
  selector for GDT: 0008
  selector for LDT: 0000
  selector for IDT: 0018
  selector for GDTPool: 0100
--- SAS Device Driver Section ---
  offset for the first bimodal dd: 0CB9
  offset for the first real mode dd: 0000
  sel for Drive Parameter Block: 04C8
  sel for ABios prot. mode CDA: 0000
  seg for ABios real mode CDA: 0000
  selector for FSC: 00C8
--- SAS Task Section ---
  selector for current PTDA: 0030
  FLAT offset for process tree head: FFF10910
  FLAT address for TCB address array: FFF06Bb6
  offset for current TCB number: FFDFFB5E
  offset for ThreadCount: FFDFFB62
--- SAS File System Section ---
  handle to MPT PTree: FE72CFBC
  selector for System File Table: 00C0
  sel. for Volume Parameter Bloc: 0788
  sel. for Current Directory Struc: 07B8
  selector for buffer segment: 00A8
--- SAS Information Segment Section ---
  selector for global info seg: 0428
  address of curtask local infoseg: 03C80000
  address of DOS task’s infoseg: FFFFFFFF
  selector for Codepage Data: 07CB
--- SAS RAS Section ---
  selector for System Trace Data Area: 04B0
  segment for System Trace Data Area: 04B0
  offset for trace event mask: 0B28
--- SAS Configuration Section ---
  offset for Device Config. Table: 0D50
--- SAS Virtual Memory Mgt. Section ---
  Flat offset of arena records: FFF13304
  Flat offset of object records: FFF1331C
  Flat offset of context records: FFF1330C
  Flat offset of kernel mte records: FFF0A891
  Flat offset of linked mte list: FFF07934
  Flat offset of page frame table: FFF11A70
  Flat offset of page range table: FFF111EC
  Flat offset of swap frame array: FFF03BAC
Flat offset of Idle Head: FFF10090
Flat offset of Free Head: FFF10080
Flat offset of Heap Array: FFF11B78
Flat offset of all mte records: FFF12E04

>>> We see this is assigned to selector c0.
>>> This is not quite the SFT but a table of selectors that point to
>>> each extent of the SFT. Each extent holds up to 500 STF entries.
>>> All the SFN’s we’re interested in are less than 500 so occupy the
>>> first extent. Note: we could have obtained the SFT selector from:
# in GDT_SFT
138:000000c0 os2krnl DOSGDTDATA:GDT_SFT
#

>>> List the table of extents:
    # dw c0:0
    00c0:00000000  0438 0000 0000 0000 0000 0000 0000 0000
    00c0:00000010  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000020  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000030  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000040  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000050  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000060  0000 0000 0000 0000 0000 0000 0000 0000
    00c0:00000070  0000 0000 0000 0000 0000 0000 0000 0000

>>> Now list the first extent:
    # dw 438:0
    0438:00000000  0000 0000 0000 0000 0001 0000 0000 0001
    0438:00000010  0000 0000 e800 0000 0000 0000 0000 0000
    0438:00000020  0000 0000 0800 c800 0000 0000 0000 0000
    0438:00000030  0000 0000 0000 0000 0000 0000 0000 0000
    0438:00000040  0000 0000 0000 0000 0000 0000 0000 0000
    0438:00000050  0000 0000 0000 0000 0000 0000 0000 0000
    0438:00000060  0000 0000 0000 0000 0000 0000 0000 0000
    0438:00000070  0000 0000 0000 0000 0000 0000 0000 0000

>>> There is an 8 byte header to each extent. It followed by one or
>>> more 131 (hex 83) byte SFT entries. The first word of the header
>>> contains the selector for the next extent.  In this case there
>>> isn’t one.
>>> To locate the SFT entry corresponding to SFN we use the formula
>>> 438:(8+(83*SFN))
>>> We can dump this out directly or by using the .D SFT command:
>>> Start by examining SFN 0077 (JFN 0006 for slot 39)
# .d sft 438:(8+(83*77))
    sf_ref_count: 0001                        sf_usercnt: 0000
    sf_flags(2): 0000:0000                  sf_usercnt: 0000
    reserved: 00                          sf_usercnt: 0000
    sf_devptr: #0000:0000                 sf_usercnt: 0000
    sf_FSC: #00c8:0008                   sf_usercnt: 0000
    sf_chain: #0000:0000                  sf_usercnt: 0000
    sf_MFT: #fe87ebf0                     sf_usercnt: 0000
    sfdFAT_firFILEclus: 57e4                sfdFAT_cluspos: 00f8
    sfdFAT_cluspos: 00f8                   sfdFAT_position: 00000000
    sfdFAT_lstclus: 0038                     sfdFAT_UID: 0000
    sfdFAT_dirsec: 00002cad                 sfdFAT_PID: 0019
    sfdFAT_dispos: 09                       sfdFAT_PDB: 0000
    sfdFAT_dirpos: 09                      sfdFAT_name: FCIDLGP DLL
    sfdFAT_file: FCLDLGP DLL                 sfdFAT_selfsfn: 0077
    sf_plock: 0000                      sfdFAT_EAHandle: 0000
    sf_NmPipeSfn: 0000                   sfdFAT_Tstamp: 00
    sf_codepage: 0000                    sf_UID: 0000
    sf_numPipeSfn: 0000                   sf_PID: 0000
    sf_position: 00000000                  sf_plock: 0000

>>> Fully qualified file system names are maintained in the Master
>>> File Table entries. Lets check out the MFT for this SFT, which
>>> is pointed to by sf_MFT.
>>> Under the Kernel debugger we could use .D MFT to format an MFT
>>> entry. Under the dump formatter .D MFT does not work correctly:
    # db %fe87ebf0
    %fe87ebf0 4b 53 45 44 0d 01 02 00 00-00 00 00 00 00 00 00 00 00
    %fe87ece0 00 00 04 0d 00 00 00 00 00 00 00 00 00 00 00 00 KSEM............
    %fe87eef0 4b 53 45 44 0d 01 02 00 00-00 00 00 00 00 00 00 00 00
    %fe87edc0 00 00 04 0d 00 00 00 00 00 00 00 00 00 00 00 00 KSEM..
The file name is at MFT+34 in the ALLSTRICT kernel and 2a in the RETAIL kernel. There are two imbedded KSEMs, which only contain the signature KSEM in the ALLSTRICT kernel, also the MFT contains the signature mF at +30 in the ALLSTRICT kernel.

The first KSEM used for serialising read/single write access to the file. The second KSEM is used for updating the cluster map.

These KSEMs can be formatted using .d KSEM

```
# .d ksem %fe87ebf0
Signature : KSEM                        Nest: 0000
Type : SHARE                    Readers: 0000
Flags : 01                PendingReaders: 0000
Owner : 0000              PendingWriters: 0000
# .d ksem %fe87ebf0+1a
Signature : KSEM                        Nest: 0000
Type : SHARE                    Readers: 0000
Flags : 01                PendingReaders: 0000
Owner : 0000              PendingWriters: 0000
#```

In this case they are unowned.

The file name in the MFT does not agree with the sf_FAT_name.

We suspect that this is not a FAT file. This can be verified by examining the file system control block entry for the FSD that’s managing this file. The FSC entry address appears in the SFT at sf_FSC. In this case it is 00c8:0000.

```
# dw c8:8
00c8:00000b6c HPFS
```

Each FSC entry is a table of far16 pointers. The first points the FSD attributes, the second to the name and the remainder are standard FSD entry points. (See OEMI IFS Documentation).

The name of this FSD is....

```
# da 840:b6c
0840:00000b6c HPFS
```

The word prefixing the file name in the MFT is the handle to the Volume Parameter Block (VPB). This also appears in the SFT under sfi_hVPB. In this instance the hVPB is 0012.

To format the VPB we need to obtain the selector for the VPB segment. N.B. this is not stored in the SAS under Volume Parameter Block. We have to locate this using:

```
# ln GDT_VPB
138:00000098 os2krnl DOSGDTDATA:GDT_VPB
```

The hVPB is an offset into the VPB segment. Format a VPB

```
# .d vpb 98:12
    vpb_flink: 0000                  vpdFAT_cluster_mask: 02
    vpb_blink: 008d                   vpdFAT_cluster_shift: 00
    vpb_ref_count: 0004              vpdFAT_first_FAT: 0000
    vpb_search_count: 0004           vpdFAT_FAT_count: 00
    vpb_first_access: 00              vpdFAT_root_entries: 0030
    vpb_signature: 444a                vpdFAT_first_sector: 06001100
    vpb_flags(2): 02:00                vpdFAT_max_cluster: 7d5c
    vpb_FSC: %0c8:0008 vpdFAT_medium: 0a
    vpb_FSC: %0c8:0008 vpdFAT_FAT_size: b213
    vpb_ID: 25be2014 vpdFAT_next_free: 00b2
    vpi_pDPB: #04c8:0038
```
Two important pieces of information in the VPB: vpi_drive and vpi_text. The drive number is the logical drive, numbering from 0. Thus 02 is drive C.

vpi_text is the volume label. In this case UNLABELED.

The VPB contains a signature which when dumped as bytes appears as JD. Each VPB is 7b bytes, the first starts at +12. Each VPB can be dumped using the formula: 98:(12+(7b*entry))

The word at +2 is a chain pointer offset to the next VPB. In this case 008d (=7b+12)

We can also obtain a link to disk device driver information from the VPB via vpi_pDPB (the disk parameter block). Under the kernel debugger this may be formatted using .D DPB, but gives erroneous results under DF.

From here we can locate the device driver header, but note that the strategy2 routine address is located from the DPB.

Returning to the JFN_table for slot 36. We now examine JFN 0004 which correlates with SFN 002a.

Dump the SFT as before:
Finding Files From Handles in a VDM

The situation in a VDM is slightly more complex, since it required the JFN to be compatible with DOS and therefore an 8-bit entity.
Furthermore, the JFN_table in DOS is traditionally imbedded or chained from the DOS PDB (or PSP). For this reason a second level of
indirection is employed.
The JFN returned from a VDM open indexes the byte array of virtual system file number (VSFNs). The VSFN ranges from 0 - 255. The high
47 (from 0xd0 though 0xfe) are used as real mode device handles. 0xff indicates an unused handle. When a VDM is created the initial PDB
contains the default array of 20 handles at label PDB_JFN_table (PDB + 0x18). This current array's far segment address is at
PDB_JFN_pointer (PDB + 0x34) and the size of the array is a word at PDB_JFN_Length (PDB + 0x32). The PDB lies on a paragraph
boundary (16-byte boundary) and its segment address is saved in the PTDA at CurrentPDB (PTDA +0x2ea) Once again the usual
precaution applies when referencing PTDA fields: their symbols are publicly defined for the current system context only. Therefore, to
reference a CurrentPDB out-of-context must be done relative to the PTDA address for that context.
These points are illustrated in the following example:
##.p 46
Slot Pid Ppid Csid Ord Sta Pri pTSD
pPTDA
pTCB
Disp SG Name
0046 001d 0007 001d 0001 blk 0200 7b732000 7b8c9a04 7b8af720 1f08 17 *vdm
##.s 46
>>> Slot 46 is a VDM but not the current context, so we locate the PDB
>>> relative to the PTDA (otherwise we could have just used
>>> dw currentpdb l1)
##dw %7b8c9a04+currentpdb-ptda_start l1
0030:0000fd16 0e01
>>> This is a segment address so use the & operator to display the
>>> PDB.
##db &e01:0
&0e01:00000000
&0e01:00000010
&0e01:00000020
&0e01:00000030
&0e01:00000040
&0e01:00000050
&0e01:00000060
&0e01:00000070
>>>
>>>
>>>
>>>
>>>
>>>

cd
28
ff
11
14
cd
20
20

20
08
ff
0e
0b
21
20
20

00
65
ff
30
00
cb
20
20

a0
07
ff
00
00
72
20
20

00
28
ff
00
00
6e
20
20

9a
08
ff
00
00
6c
20
20

f0
99
ff
40
00
20
20
20

fe-1d
0d-d1
ff-ff
09-ff
00-00
2d-20
20-00
20-00

f0
d1
ff
ff
00
4e
00
00

f5
d1
ff
ff
00
6f
00
00

01
d0
ff
ff
00
74
00
00

99
d2
d7
00
00
00
00
e7

0d
ff
00
00
00
20
20
08

08
ff
e4
00
00
20
20
50

02
ff
03
00
00
20
20
03

M . ..p~.pu.....
(.e.(...QQQPR...
............W.d.
..0...@.........
................
M!Krnl - Not.
.....
....g.P.

Word at PDB+0x32 is 0030. This is the number of file handles
supported in this VDM. The default is 0014. So the PDB_JFN_table
has been expanded.
Far pointer at PDB+34 is &0940:0000. This is the current
PDB_JFN_table address. (By default this would have pointed to
PDB+0x18, but the table has been expanded.)

>>> Now dump the current PDB_JFN_table.
##db &940:0
&0940:00000000
&0940:00000010
&0940:00000020
&0940:00000030
&0940:00000040
&0940:00000050
&0940:00000060
&0940:00000070
>>>
>>>
>>>
>>>
>>>
>>>
>>>

d1
0b
ff
4d
00
00
00
00

d1
ff
ff
00
00
00
00
00

d1
ff
ff
00
00
00
00
00

d0
ff
ff
08
00
00
00
00

d2
ff
ff
00
00
00
00
00

00
ff
ff
00
00
00
00
00

01
ff
ff
00
00
00
00
00

02-03
ff-ff
ff-ff
00-00
00-00
00-00
00-00
00-00

04
ff
ff
00
00
00
00
00

05
ff
ff
00
00
00
00
00

06
ff
ff
00
00
00
00
00

07
ff
ff
00
00
00
00
00

08
ff
ff
00
00
00
00
00

09
ff
ff
00
00
00
00
00

ff
ff
ff
00
00
00
00
00

QQQPR...........
................
................
M...............
................
................
................
................

JFNs 0 - 4 correspond to VSFNs d1, d1, d1, d0 and d2. Each of these
is greater than 0xcf and therefore a real mode device handle
and not managed by the protect mode file system.
JFN 5 is the first open file in this VDM. It has VSFN 00. This
may be used as an index into the protect mode JFN_table to
find the corresponding SFT, MFT and file name. The technique from
this point is the same as in the preceding section.

>>> Dump the JFN_table from the PTDA.
##dw %7b8c9a04 jfn_ptable-ptda_start l2
0030:0000ffbc 0000 1ea8
>>> We are no longer based on selector 30 so the JFN table has been
>>> expanded. Now dump the current table...
##dw #1ea8:0
1ea8:00000000

006a 0069 0075 008c 008b 005e 0089 0088


Finding Handles From File Names

File system names are recorded in the MFT control block. Each MFT has a handle, which is its linear address and a key which is the concatenation of the hVPB with the file name considered as a string of bytes. The MFT keys are managed in a Patricia Tree structure similar to that described by Knuth in The Art of computer Programming, Volume 3, Sorting and Searching Algorithms. However, note that the implementation of the PTree in OS/2 is slightly modified from the Knuth treatment.

The SAS gives us the address of the header node for the MFT PTree. The header node points to the first PTree entry. Each entry comprises a bit index, a key length, a left pointer, a right pointer and an MFT handle. The bit-index is used to specify the bit in the key to be tested. If the bit 0 then the left pointer is taken, otherwise the right pointer is taken. When the selected pointer points back to the same PTree entry then the search stops and the required MFT is found from the MFT handle. The bit index count the bits of each byte of the key from left to right.

This technique is now illustrated in the following example:
>>> Who's got C:\OS2\HELP\HMHELP.HLP open?

>>> First look through VPBs to find hVPB for C:
>>> Find the VPB segment and chain through them starting with the
>>> first at offset 0x12.

# ln gdt_vpb
0138:00000098 OS2KRNL GDT_VPB
# .d vpb 98:12

vpb_flink: 0000 vpdFAT_cluster_mask: 02
vpb_blink: 008d vpdFAT_cluster_shift: 00
vpb_ref_count: 0057 vpdFAT_first_FAT: 0000
vpb_search_count: 0004 vpdFAT_FAT_count: 00
vpb_first_access: 00 vpdFAT_root_entries: 0030
vpb_signature: 444a vpdFAT_first_sector: 06001100

>>> We get lucky the first time. This VPD is for drive 2, that is C:
>>> So the hVPB=0012 (i.e the offset into the VPD segment).

>>> We now need to form the MFT key for the file name we wish to
>>> look up. So convert the file name to ASCII and concatenate to the
>>> hVPB (as a byte pair, that is, reversed)
>>> C  :  \  O  S  2  \  H  E  L  P  \  H  M  H  E  L  P  .  H  L  P
>>> Locate the MFT PTree head from the SAS - in a dump use .A
>>> otherwise unravel the SAS from selector 70

# .a
--- SAS Base Section ---
SAS signature: SAS
offset to tables section: 0016
FLAT selector for kernel data: 0168
offset to configuration section: 001E
offset to device driver section: 0020
offset to Virtual Memory section: 002C
offset to Tasking section: 005C
offset to RAS section: 006E
offset to File System section: 0074
offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
selector for GDT: 0008
selector for LDT: 0000
selector for IDT: 0018
selector for GDTPool: 0100
--- SAS Device Driver Section ---
offset for the first bimodal dd: 0CB9
offset for the first real mode dd: 0000
set for Drive Parameter Block: 04C8
set for ABIOS prot. mode CDA: 0000
seg for ABIOS real mode CDA: 0000
selector for FSC: 00C8
--- SAS Task Section ---
selector for current PTDA: 0030
FLAT offset for process tree head: FFF10910
FLAT address for TCB address array: FFP06B6B
offset for current TCB number: FFDPFFB5E
offset for ThreadCount: FFPFFB62
--- SAS File System Section ---
handle to MFT PTree: FE72CFBC
selector for System File Table: 00C0
sel. for Volume Parameter Bloc: 0788
sel. for Current Directory Struc: 07B8
selector for buffer segment: 00A8
--- SAS Information Segment Section ---
selector for global info seg: 0428
address of curtask local infoseg: 03C80000
address of DOS task's infoseg: FFFFFFFF
selector for Codepage Data: 07CB
--- SAS RAS Section ---
selector for System Trace Data Area: 04B0
segment for System Trace Data Area: 04B0
offset for trace event mask: 0B28
--- SAS Configuration Section ---
offset for Device Config. Table: D050
--- SAS Virtual Memory Mgt. Section ---
Flat offset of arena records: FFF13304
Flat offset of object records: FFF1331C
Flat offset of context records: FFF1330C
Flat offset of kernel mte records: FFF0A891
Flat offset of linked mte list: FFF07934
Flat offset of page frame table: FFF11A70
Flat offset of page range table: FFF111EC
Flat offset of swap frame array: FFF03BAC
Flat offset of Idle Head: FFF000B0
Flat offset of Free Head: FFF10080
Flat offset of Heap Array: FFF11B78
Flat offset of all mte records: FFF12E04

>>> MFT Ptree is at %fe72cfbc
>>> each entry including the header has the following format:
>>> +0 W Bit index
>>> +2 W key length
>>> +4 D left link
>>> +8 D right link
>>> +c D MFT handle (MFT pointer)
>>> dump the header and the first entry pointed to by the left link
# dd %FE72CFBC l4
%fe72cfbc  ffffffff fe867f10 fe72cfbc 00000000
# dd %FE867F10 l4
%fe867f10  00100000 fe861454 fe861470 fe721a04
>>>        ----.... -------- -------- --------
>>>        Kl  BI    left     right    MFT
>>> Note the word reversal of the Bit index and the Key length because
>>> we dumped double-words.
>>> BI tells us to test bit 0 of the key (numbering from the left
>>> starting with 0). 12 00 .... .... = 0001 0010 0000 0000 .... ....
>>> Bit zero is 0 so take the left link.
# dd %FE861454 l4
%fe861454  00100001 fe73d194 fe845370 fe72196c
>>> Now test bit 1. This is still zero. Again take the left link.
# dd %FE73D194 l4
%fe73d194  00190003 fe72cf3c fe87ec34 fe72dea4
>>> Now test bit 3. This is 1 so take the right link.
# dd %FE87E3C4 l4
%fe87ec34  000b0029 fe87ebdc fe834308 fe87ebf0
>>> Now test bit 29. .... 4f .... = 0100 1111
>>> This is 1 so take the right link.
# dd %FE834308 l4
%fe834308  0019002b fe869f30 fe834254 fe834274
>>> Test bit 2b. This is 0. Turn left.
# dd %FE869F30 l4
%fe869f30  0017002c fe885ac4 fe87ec90 fe869ee0
>>> Test bit 2c. This is 1. Turn right.
# dd %FE87EC90 l4
%fe87ec90  00Df002d fe87ec90 fe834ac8 fe87ec48
>>> Test bit 2d. This is 1. Turn right.
# dd %FE834AC8 l4
>>> Test bit 44. ....5c.... = 0101 1100.  
>>> This is 1. Turn right.
# dd %FE724fe4 l4
%fe724fe4 001b004b fe801414 fe862de8 fe722fac
>>> Test bit 4b. ......48...... = 0100 1000  
>>> This is 0. Turn left.
# dd %FE801414 l4
%fe801414 0017004c fe801d90 fe7cef84 fe7dffb0
>>> Test bit 4c. This is 1. Turn right.
# dd %FE7cef84 l4
%fe7cef84 00180073 fe7cef84 fe801414 fe7cef30
>>> Test bit 73. ......48.... = 0100 1000  
>>> This is zero and the left link points to the same node.  
>>> Therefore the search ends and we should have found the MFT  
>>> for our file name. Dump the MFT to check ......
# db %FE7cef30 150
%fe7cef30 4b 53 45 4d 01 02 00 00-00 00 00 00 00 00 00 00 00 00 00 00 KSEM..........  
%fe7cef40 00 00 00 28 31 38 04 00 00-00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..........  
%fe7cef60 46 12 00 43 3a 5c 4f-53 32 5c 48 45 4c 50 5c 69 03 00 00 .................  
%fe7cef70 48 4d 48 45 4c 50 2e 48-4c 50 00 00 16 e6 7c fe 8b32\HELP\...
>>> The MFT + 0x22 points is the SFT segment’s offset. So dump the  
>>> SFT ....
# ln gdt_sft
0138:000000c0 OS2KRNL GDT_SFT  
dw c0:011  
#c0:00000000 0438  
# .d sft 438:3128
    sf_ref_count: 0001  
    sf_usercnt: 0000  
    reserved: 00  
    sf_flags(2): 0000:0000  
    sf_devptr: #0000:0000  
    sf_FSC: #00c8:0008  
    sf_chain: #0438:33b7  
    sf_MFT: fe7cef30  
    sfdFAT_firFILEclus: 3344  
    sfdFAT_cluspos: 0f10  
    sfdFAT_lstclus: 0000  
    sfdFAT_dirsec: 00000000  
    sfdFAT_EAHandle: 0000  
    sf_plock: 0000  
    sf_codepage: 0000  
    sfdFAT_name: sfi_selfsfn: 0060  
    sf_MFT: fe7cef30  
    sf_plock: 0000  
    sf_ref_count: 0001  
    sf_usercnt: 0000  
    reserved: 00  
    sf_FSC: #00c8:0008  
    sf_chain: #0438:2d10  
    sf_MFT: fe7cef30  
    sfdFAT_firFILEclus: 284a  
    sfdFAT_cluspos: 0f10  
    sfdFAT_lstclus: 0000  
    sfdFAT_dirsec: 00000000  
    sfdFAT_name: sfi_selfsfn: 0065  
>>> sfi_PID tells us PID 12 has opened this file. But the  
>>> sf_chain is not zero, so other processes have also opened  
>>> this file. Follow the sf_chain ......
The Record Lock Record

In this example we investigate the RLR and how it records a locked range within a file. We will also see how the Block-Id of a thread waiting for access to a locked file range directly leads to discovery of the RLR.

We introduce the RLR by showing its relationship to other file-system control blocks in the following diagram. This depicts the following situation:

- Three processes have opened the same file, in the order process 1, 2, then 3.
- The MFT heads the chain of SFTs, each representing an open instance of the same file. The MFT points to the most recent SFT.
- Process 1 and process 2 have each locked a range within the same file. RLRs 1 and 2 correspond to process 1 and 2.
- The MFT heads the chain of RLRs starting with the most recent. The pointer from the MFT (lptr) is the offset within the RLR segment.

---

>>> In all, PIDs 0012, 000c, 000b and 000d have opened
>>> C:\OS2\HELP\HMHELP.HLP

The Record Lock Record
Shared File with 2 Locked Ranges
A Hang Problem Involving Locked Records

>>> Problem: Program "Pain" running in Slot 4d is hung with a blank screen. Everything else in the system seems OK. Mouse moves, we can change focus and so on.

>>> Lets take a look at slot 4d.

```shell
# .p 4d
 Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
004d# 001c 001b 001c 0001 blk 0300 ab80f000 ab99e220 ab980620 1e60 1d PAIN
```

>>> Blocked!

>>> We can approach this two ways:

>>> 1) Take a look at what the application did

>>> 2) Take a look at the BlockId and try see how far the system got

>>> Looking at the application we examine its registers and determine what API it called to cause it to block.

```shell
# .s 4d
Current slot number: 004d
```

```plaintext
%00010171 e8508d45e0 call 7e0468ec6
%00010176 50 push eax
%00010177 ff75f8 push dword ptr [ebp-08]
%0001017a b005 mov al,05
%0001017c e8c762f81b call %1bf96448
%00010181 83c414 add esp,+14
%00010184 0bc0 or eax,eax
%00010188 7404 jz %00010168
%0001018a 8bc0 mov eax,eax
%0001018c b858000200 mov eax,00020058
%00010191 e8fa0000 call %00010290
```

```shell
# u %eip-10
%1bf96448 DOSCALL1 DOS32SETFILELOCKS
```

>>> The last call was to DosSetFileLocks and we haven't returned. If we want any more information we have to analyse the BlockId.

```shell
# .pb#
```

```plaintext
Slot Sta BlockID Name Type Addr Symbol
004d# blk 00b80029 PAIN
```

```shell
*har par cpq va flg next prev link hash hob hal
00dd %eaf0308 00000010 kaa7a3000 129 00dc 00de 0000 00cb 00ea 0000 sel=00b8
hob har hob nat flgs own hmt e sown cnt 1 st xf
00ea 00dd 0000 124 ff47 0000 0000 00 00 00 00 00 fareclok
```
>>> Seems to be blocked on a File System Record Lock (RLR).
>>> This implies that someone else has already locked a
>>> conflicting record range.

>>> We need to dump the RLR and locate the System File Table Entry
>>> associated with it. The BlockId is the address of the RLR.

\# dw 0b8:29
00b8:00000029  0000 0000 0020 0000 002f 0000 526b 00d0
00b8:00000039  0000 0018 0000 0002 0000 2000 0000 2000
00b8:00000049  0000 9806 29ab 1c00 0300 0000 006e
00b8:00000059  0000 0000 0000 0000 0000 0000 0000 0000
00b8:00000069  0000 0000 8500 0000 0000 0000 0000 0000
00b8:00000079  0000 0000 0000 0000 0000 0000 0000 0000
00b8:00000089  0000 0000 0000 0000 0000 0000 0000 0000
00b8:00000099  0000 0000 0000 0000 0000 0000 0000 0000

>>> RLR+c is a far16 pointer to the associated System File Table entry
>>> (SFT).

>>> RLR+4 and RLR+8 are the range of bytes locked. Offset +20 - +2f
>>> has been locked.

>>> We now format the SFT for the process that locked this range:

\# .d sft d0:526b

sf_ref_count: 0001
sf_usercnt: 0000
reserved: 00
sf_flags(2): 0040:0000
sf_devptr: #0000:04e0
sf_FSC: #0000:ff40
sf_chains: #0000:0000
sf_FFT: fe87ca0c
sf_FAT_firFILEclus: 0197
sf_FAT_lastclus: 0197
sf_FAT_dirsec: 0000009f
sf_FAT_dirpos: 0a
sf_FAT_name: VIN
sf_FAT_EAHandle: 0000
sf_FAT_PipeSfn: 0000
sf_FAT_codepage: 0000

\# .d mft % fe87ca0c

mft_ksem:
Signature     : KSEM
Type          : SHARE
Flags         : 01
Owner         : 0000
mft_lptr: 0029
mft_sptr: 00d0:5600
mft_pMap: 00000000
mft_hwvpb: 466d
mft_opflags: 0000
mft_flags: 0000

\# .p

Slot   Pid   Ppid  Csid   Ord  Sta    Pri  pTSD     pPTDA     pTCB      Disp SG Name
0001   0001 0000 0000  0001 blk  0100 ffe3a000 ffe3ca00 ffe3c800 1e7c 00 *ager
0002   0001 0000 0000  0002 blk  0200 ab779000 ffe3ca00 ab977020 1f3c 00 *tsd
0003   0001 0000 0000  0003 blk  0200 ab77b000 ffe3ca00 ab977220 1f50 00 *ctxh

\# .p

Slot   Pid   Ppid  Csid   Ord  Sta    Pri  pTSD     pPTDA     pTCB      Disp SG Name
0001   0001 0000 0000  0001 blk  0100 ffe3a000 ffe3ca00 ffe3c800 1e7c 00 *ager
0002   0001 0000 0000  0002 blk  0200 ab779000 ffe3ca00 ab977020 1f3c 00 *tsd
0003   0001 0000 0000  0003 blk  0200 ab77b000 ffe3ca00 ab977220 1f50 00 *ctxh
Pid 18 is evidently FROMAGE.
FROMAGE has the VIN and PAIN wants it!

We had better find out why FROMAGE has blocked.

Current slot number: 0049

eax=00000001 ebx=00020366 ecx=1bf90000 edx=00020004 esi=13fa0000 edi=13fa1052 
eip=0000a0c3 esp=0000324a ebp=00023270 iopl=2 -- -- nv up ei ng nz ac pe cy cs=ddf7 ss=0017 ds=9fe7 es=9fe7 fs=150b gs=0000 cr2=00000000 cr3=00010900 

Invalid linear address: ddf7:0000a0c3

Looking at this from the application aspect may be difficult since some of the code has been paged out (The invalid linear address message).

The near symbol gives a clue. We could try unwinding the stack and hope that the stack is still paged in.

%23270 looks like a candidate. Let's unassemble the return address to see if it makes sense.

This is going to be haphazard. Evidently the code we are currently executing is not using EBP as a stack frame pointer. All we can do is scan through the stack looking for a likely stack frame or a return address to user code.

%232f0 looks like a candidate. Let's unassemble the return address to see if it makes sense.

%2350 looks like a candidate. Let's unassemble the return address to see if it makes sense.

The most up-to-date status indicator for this process.

Slot Sta BlockID Name Type Addr Symbol
0049 blk 05100604 FROMAGE
This blockid points to data within the kernel resident read/write heap. Heap blocks have headers that tell us more about the user of the data. The data portion of a help block is usually mapped to a GDT selector. In this example, selector 510. 510:0 should be the address of the beginning of the data and therefore point just after the end of the header. We look at the data before 510:0 to see the heap block header.

>>> This tells us selector 510 was allocated by, or is part of the 4th device driver to initialise. Listing the physical device driver MTBs will find this. They are listed last initialised first.

Note: frequently we find that dd16 is the owner. This refers to all device drivers from the 16th and subsequent. The first 15 device drivers to initialise are assigned unique owner ids from dd1 to dd15, where the numbers are in decimal.
hmte=0123 pmte=%fe83df4c mflags=8008f1c9 c:\os2\com.sys
hmte=0121 pmte=%fe839e64 mflags=8008f1c9 h:\os2\boot\mouse.sys
hmte=011d pmte=%fe83afdc mflags=8008f1c9 h:\os2\boot\os2cdrom.dmd
hmte=0111 pmte=%fe83af88 mflags=8008f1c9 h:\os2\boot\pmdd.sys
hmte=0087 pmte=%fe839fdc mflags=0008f1c9 h:\os2\boot\dos.sys
hmte=0089 pmte=%fe839f80 mflags=8008f1c9 h:\os2\boot\testcfg.sys
hmte=0103 pmte=%fe71095c mflags=8008f1c9 i:\brew\os20menu.sys
hmte=00e2 pmte=%fe6f6fd64 mflags=8008e1c9 h:\os2scsi.dmd
hmte=00de pmte=%fe6f66f0 mflags=8008e1c9 h:\xdfloppy.flt
hmte=009a pmte=%fe6f6fb0 mflags=8008e1c9 h:\fd16-700.add
hmte=00a7 pmte=%fe6f6f3c mflags=8008e1c9 h:\ibms506.add
hmte=00c9 pmte=%fe6f6f0 mflags=8008e1c9 h:\print01.sys
hmte=009b pmte=%fe6f6f0 mflags=8008e1c9 h:\ibmkbd.sys
hmte=009a pmte=%fe6f6f0 mflags=8008e1c9 h:\kbdbase.sys
hmte=0099 pmte=%fe6f6f0 mflags=8008e1c9 h:\screen01.sys
hmte=0098 pmte=%fe6f6f0 mflags=8008e1c9 h:\clock01.sys
hmte=0096 pmte=%fe6f6f0 mflags=8008e1c9 h:\resource.sys

>>> Counting backwards, the 4th device driver is kbdbase.sys.

>>> We dump its object table.

# .lmo 9a
hmte=009a pmte=%fe6f6fd64 mflags=8008e1c9 h:\kbdbase.sys
seg sect psiz vsiz hob sel flags
0001 001 158c 170e 0000 0510 8c41 data prel
0002 000c 3270 3270 0000 0518 8d60 code shr prel rel
0003 0026 1987 1988 0000 0520 8d60 code shr prel rel
0004 0033 0743 0744 0000 0528 8d60 code shr prel rel

>>> Selector 510 is indeed the first data selector of kbdbase.sys and
>>> will contain the device driver header at offset +0

# .d dev 510:0
DevNext: 0500:0000
DevAttr: c981
DevStrat: 0000
DevInt: 2d29
DevName: KBD$
DevProtCS: 0518
DevProtDS: 0510
DevRealCS: 0000
DevRealDS: 0000

>>> We conclude that FROMAGE is waiting for the device driver to
>>> respond to the DosRead - i.e. A keyboard interrupt

--------------------------------------------
Exploring Memory Management

This section gives a basic overview of memory management, and shows how to answer the following questions:

1. Who owns and who allocated virtual memory?
2. How to correlate named memory with its address and vice versa
3. How memory aliasing works.

If the reader is unfamiliar with this subject then these sections should be read in order.

Who Owns Virtual Memory and Who Allocated it?
In this section we take a look at the primary system structures used in memory management and how they are located using the Dump Formatter and Kernel Debugger. These structures are:

- The memory arena record (VMAR)
- The memory arena header record (VMAH)
- The memory object record (VMOB)
- The memory context record (VMCO)

The examples worked in this section illustrate:

- how to find all memory allocations made by a given process and what executable made the allocation.
- how to determine ownership of non-system memory.
- the use of memory objects, pseudo-objects and system objects.

Memory allocations have many attributes, included among which are:

- data or content
- location or address
- size
- ownership
- requestor

The composite set of attributes associated with a memory allocation is referred to as a memory object. OS/2's virtual memory manager tracks memory objects using arena, object and context records.

We start by looking at the arena record, which is used to record virtual address assignments to memory objects.

The entire system address space of 4 gigabytes is partitioned into three types of memory arena:

**System Arena**

This is the range of virtual addresses where system information and ring0 code executes. Typically device drivers, file system drivers and the OS2 Kernel executes and uses data assigned to the System Arena. There is just one instance of the System Arena. It is assigned the virtual address range from 512 Mb to 4 Gb.

**Shared Arena**

This is the range of virtual addresses assigned to shared objects. Shared data objects come in two varieties:

- Global data: Such objects exist as unique entities. Their address range and data content are common to all accessing processes. This is achieved by using common page tables in all processes.
- Instance data: Such objects share the same address range, but exist as distinct data instances in each accessing process. Page table entries for instance data are specific to each process.

Code objects from DLL modules are also consigned to the shared arena.

In general processes are not given automatic access to instance or global data. Access is granted either implicitly by the system loader because of calls to other DLLs or explicitly by use of the DosGiveXxxx and DosGetXxxx set of APIs.

There is just one shared arena, which reserves initially virtual memory addresses from 304Mb to 512Mb. This may be expanded by lowering the lower boundary. The current address range assigned to the shared arena is managed by a special arena record called the boundary sentinel arena record.

**Private Arena**

This is the range of virtual addresses used to map objects that are unique to each process. A private arena therefore exists for each process. In general the page tables of each private arena will map to unique real storage frames. An exception to this is with code objects. Since code segments are always read-only then if more than one process is running the same executable module their page tables will map to a common set of real storage frames for the code segments of the executable module.
Private arenas are assigned an initial address range from 64k to 64M. This may be expanded upwards as more memory is allocated. The current size of a private arena is tracked by a special arena record called the sentinel arena record.

The private arena upper boundary and shared arena lower boundary may grow towards each other but not overlap.

These worked examples now follow:

Exploring arena records
Exploring object records
Finding who owns memory

--------------------------------------------

Exploring Arena Records

The following example illustrates the use of arena records:

>>> We start by asking the question: what ranges of addresses are currently allocated in the private arena of the process that's running the IPFC compiler.

>>> List all processes to find the one of interest

```
# .p

Slot  Pid  Ppid  Csid  Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG  Name
0001  0001 0000 0000 0001 blk 0100 ffe3a000 ffe3c7d4 ffe3c61c 1e7c 00  ager
0002  0001 0000 0000 0002 blk 0200 7b7aa000 ffe3c7d4 7b9a8020 1f3c 00  *tsd
0003  0001 0000 0000 0003 blk 0300 7b7b0000 ffe3c7d4 7b9a81d8 1f50 00  *ctxh
0004  0001 0000 0000 0004 blk 0400 7b7b2000 ffe3c7d4 7b9a8548 1f20 00  *lazyw
0005  0001 0000 0000 0005 blk 0500 7b7b4000 ffe3c7d4 7b9a8700 1f3c 00  *asyncr
0006  0001 0000 0000 0006 blk 0600 7b7b6000 ffe3c7d4 7b9a88b8 1ecc 01  PMSHL32
0007  0001 0000 0000 0007 blk 0700 7b7b8000 7b9c4020 7b9a8a70 1eb8 01  PMSHL32
0008  0001 0000 0000 0008 blk 0800 7b7b8000 7b9c4020 7b9a8c28      00  LOGDAEM
0009  0001 0000 0000 0009 blk 0900 7b7b2000 7b9c4020 7b9a90bc      01  PMSHL32
0010  0001 0000 0000 0010 blk 0a00 7b7b4000 7b9c4020 7b9a92ec      01  PMSHL32
0011  0001 0000 0000 0011 blk 0b00 7b7b6000 7b9c4020 7b9a94d0      01  PMSHL32
0012  0001 0000 0000 0012 blk 0c00 7b7b8000 7b9c4020 7b9a96f0      01  PMSHL32
0013  0001 0000 0000 0013 blk 0d00 7b7b2000 7b9c4020 7b9a9890      01  PMSHL32
```

Finding who owns memory

```
From the name printed in the right hand column we see that slot 39 is the one of interest.

Imbedded in each PTDA at offset +0x40 is the VMAH that heads the private arena. From the VMAH we can obtain the pointer to the sentinel area record.

Dump out the VMAH for slot 39 using the pPTDA address from the .p command output...

The third double word (feb24cae) is the address of the sentinel record. To format this using the .MA command we need to determine the handle for this record. Arena records are organised in a table of 0x16 byte length entries. Their handles are their corresponding table entry number. The address of the first arena record is located at symbol _parvmone...

Arena record 1 is located at %feb1f020. We wish to determine the handle for the sentinel, whose address is %feb24cae. We use the hex calculator facility of the Dump Formatter/Kernel Debugger thus...

The handle we require is 436. We can now format the sentinel for slot 39....

Note the max=%04000000 to the right indicating the current private arena maximum address is 64M - 1 and incidentally distinguishing this as a sentinel or boundary sentinel arena record.

Note also that this is merely a boundary marker and not an indication of which addresses within the private arena have been allocated.

Regular arena record are chained to the sentinel in a circular double linked list using the 'next' and 'prev' pointers.

We can format the entire chain using .MAL (or .MAR) but we have to break in using Ctrl-C to stop the chain endlessly traversing the circular chain.
Each regular private arena record is distinguished by the appearance
hptda=nnn to the right of each line. This is the handle of the PTDA
of the process to which the arena record belongs. Each of the hptda
values is 50c indicating each of regular arena records above belongs
to the same process. More on the hptda later.

Each regular arena represents the address range reserved for a
memory object. cpg is the size reservation in pages, but note
that this is only an address space reservation, not necessarily what
is currently committed. Most objects reserve 0x10
pages or 64K, which corresponds to the maximum 16-bit segment size.

va shows the start address of each memory object.
By examining va and cpg we can see that the minimum and maximum
addresses allocated in the private arena of slot 39 is %10000 and
%36ffff (=%360000 + 0x10 pages -1). We can also see that this
allocation is contiguous and therefore the total allocated private
arena virtual address space is 0x360000 bytes or 3.375M

The VMAH records the minimum and maximum +1 allocated addresses
at +0x20 and +0x24, but the allocation might be sparse so the VMAH
does not indicate directly the total memory in use.

We now move onto the shared arena.

The link field of each sentinel points to the boundary sentinel

Once again each regular arena record in the shared arena is linked
in a circular double linked list. This time we enter the chain from
the boundary sentinel next and prev fields.
there are two types of regular arena record that appear in the  
shared arena. these are distinguished by the right-hand column:  
\[ hco = \text{nnnnn} \]  
the first type is global shared data. the hco is the context record  
handle, which will be discussed later.  
the second type represents instance data. both of these will be  
looked at in more detail in the next section.  

finally we look at the system arena. the sentinel for the system  
arena is \( \text{har} = 4 \). once again each regular arena record is linked  
in a circular double-linked list.
Exploring Object Records

We now explore the memory object record (VMOB) and the .MO command.

>>> There are two types of object managed by object records:
>>> pseudo-objects
>>> non-pseudo-object

>>> Non-pseudo-objects have an associated arena record. These are by far the most common type of memory object. They include code and data segments of application and system code. However, there is a draw-back in that arena records deal with page size quantities. For certain system control blocks it is useful to have distinct objects associated with each instance of them. But these objects are generally very much smaller than a page. To overcome these difficulties such objects are sub-allocated from the system heap and given an object type of pseudo-object. They have no associated arena record.

>>> We list a few pseudo-objects. Note the 'p' parameter of .mo to do this.

# .mop
hob       va     flgs own  hmte  sown,cnt lt st xf
0004  %fff13238  8000 ffe1 0000  0000 00  00 00 00 vmah
0005  %fff13190  8000 ffe1 0000  0000 00  00 00 00 vmah
0006  %fff0a891  8000 ffa6 0000  0000 00  00 00 00 mte
doscalls.dll
0072  %ffe3c7d4  8000 ffc6 0000  0000 00  00 00 00 ptda  0001 *sysinit
007a  %fff0b3fa  8000 ffa6 0000  0000 00  00 00 00 mte
mvmd.dll
007b  %fff0b26b  8000 ffa6 0000  0000 00  00 00 00 mte
fshelper.dll
007d  %fe720f60  8000 ffa6 0000  0000 00  00 00 00 mte
a:mini_fsd.fsd
0086  %fe861f30  8000 ffa6 0000  0000 00  00 00 00 mte
cicmd.sys
0087  %fe861f58  8000 ffa6 0000  0000 00  00 00 00 mte
citestcfg.sys
0088  %fe860f9c  8000 ffa6 0000  0000 00  00 00 00 mte
cprint01.sys
0089  %fe721f1c  8000 ffa6 0000  0000 00  00 00 00 mte
ciclock01.sys
0099  %fe7246bc  8000 ffa6 0000  0000 00  00 00 00 mte
cid:pmshell.exe
0091  %7b9c484c  8000 ffc6 ff79  0000 00  00 00 00 ptda 0003 c:pmshell.exe
0096  %fe721fbb  8000 ffa6 0000  0000 00  00 00 00 mte
ciclock01.sys
009f  %fe724f84  8000 ffa6 0000  0000 00  00 00 00 mte
cibm1flpy.add
00a1  %fe725f8c  8000 ffa6 0000  0000 00  00 00 00 mte
cibm1s506.add

>>> Pseudo-objects apply to four types of control block but in general we will only be concerned with the PTDA and the MTE. The pseudo-object record is distinguished by the presence of the 'va' field. The control block name is shown in column 11.

>>> The 'va' field gives the address of the object itself. In this case it's a PTDA address. We can find the thread slots which correspond to this PTDA either by using .P and looking for a match in the pPTDA field or directly:

# dw %7b9c484c+pid-ptda_start l1
%7b9c5042  0003

>>> This is the pid. Note .mo 91 extracts this for us - the pid appears...
>>> after 'ptda's'

```bash
# dd %7b9c484c+ptda_pTCBHead-ptda_start l1
%7b9c486c 7b9a8a70
# dw 7b9a8a70 12
%7b9a8a70 0001 0008
>>> This is the head of the TCB tree for pid 3.
>>> Words 0 and 1 of the TCB contain the thread ordinal and its slot
>>> number. This is tid 1 in slot 8.
# .p8
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD    pPTDA    pTCB    Disp  SG  Name
0008  0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01  PMSHL32

>>> The PTDA for slot 8 has pid 3, is at %7b9c484c which is hob 91.
>>> The handle of the PTDA (hptda) is defined to be the hob of the
>>> object it occupies. Thus this ptda is also identified by hptda=91
>>> The other most frequently encountered pseudo-object is the mte.

hob       va     flgs own  hmte  sown,cnt lt st xf
0193  %fe722dec  8000  ffa6 0000 0000 00 00 00 00 00 00 mte  c:\pmshell.exe

>>> The MTE represents a loaded module. In this case the MTE control
>>> block is located at %fe722dec and is assigned the mte handle of its
>>> hob. In this case the MTE at %fe722dec is also referred to as
>>> hmte=193. The .LM command will respond to either hmte or MTE address
>>> and format the MTE for us ...
# .lm 193
hmte=0193 pmte=%fe722dec mflags=84903150 c:\os2\pmshell.exe

>>> .MO extracts the module name from the MTE and displays this to the
>>> right of 'mte'
>>> In each object record is the 'own' and 'hmte' fields. These are used
>>> to attribute ownership and associate a module or part of the system
>>> that was involved with the allocation request. In many cases
>>> these fields contain hobs of related objects. In some cases
>>> attribution needs to be made to a system resource. For this a
>>> number of generic system object Ids have been defined. They all
>>> are greater than 0xff00. .MO will translate system object Ids into
>>> a more meaningful text string. For example: in hob 193 the
>>> owner id ffa6
# .mo ffa6
ffa6 ldrmte
>>> Similarly in hob 91 the owner is ffcb
# .mo ffcb
ffcb ptda

>>> Both of these give an indication of the type of system object.
>>> In the first case a load MTE, in the second a PTDA. The 'own' and
>>> 'hmte' interpretation is used to form the description that appears
>>> to the right of each .MO line.

>>> We now turn our attention to non-pseudo objects or normal
>>> memory objects:

# .mon
hob   har hobnxt flgs own  hmte  sown,cnt lt  st  xf
0001 0001  fec8 0000  fff1 0000 0000 00 00 00 00 vmob
0002 0002  fec8 0000  ffe3 0000 0000 00 00 00 00 vmar
0003 0003  fec8 0000  ffec 0000 0000 00 00 00 00 vmkrhw
0007 0006 0000  ff6d 0000 0000 0000 00 00 00 00 doship
0008 0007 0000  ff9a 0000 0000 0000 00 00 00 00 os2knl
0009 0008 0000  ff9a 0006 0000 0000 00 00 00 00 os2knl
000a 0009 0000  ff9a 0006 0000 0000 00 00 00 00 os2knl
000b 000a 0000  ff9a 0006 0000 0000 00 00 00 00 os2knl
000d 000c 0000 3225 0006 0000 0000 00 00 00 00 os2knl
000e 000d 0000 0000 0000 0000 00 00 00 00 os2knl
000f 000e 0000 0000 0000 0000 00 00 00 00 os2knl
0010 0087 0000 0000 0000 0000 0000 00 00 00 00 priv 0003 c:\pmshell.exe
0011 0010 0000 0000 0000 0000 0000 00 00 00 00 os2knl
0012 0011 0000 0000 0000 0000 0000 00 00 00 00 os2knl
0013 0012 0000 0000 0000 0000 0000 00 00 00 00 os2knl
>>> Many of these objects have a system ID owners but those of current
>>> interest are objects allocated within the shared and private arenas
>>> by application programs.

>>> Private arena private data:
>>> --------------------------

>>> We start by examining hob 10 in more detail.

>>> We list the object and its associated arena record using the 'c'
>>> parameter of .MO
# .moc 10
*har par cpg va flg next prev link hash hob hal
0010 00f0 0000 0012 ffc4 0000 0000 00 00 00 00 smdfh
0018 00f0 0000 0000 ffc5 0100 00 00 00 00 give
0019 00f5 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001b 00f4 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001c 00f3 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001d 00f2 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12

>>> We can tell from its location (%70000) that this is a private
>>> arena address in process hptda=91. The 'own' field of the object
>>> record is also hob 91 which again implies an object owned by the
>>> process. That means the object is either a dynamic allocation or

>>> Many of these objects have a system ID owners but those of current
>>> interest are objects allocated within the shared and private arenas
>>> by application programs.

>>> Private arena private data:
>>> --------------------------

>>> We start by examining hob 10 in more detail.

>>> We list the object and its associated arena record using the 'c'
>>> parameter of .MO
# .moc 10
*har par cpg va flg next prev link hash hob hal
0010 00f0 0000 0012 ffc4 0000 0000 00 00 00 00 smdfh
0018 00f0 0000 0000 ffc5 0100 00 00 00 00 give
0019 00f5 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001b 00f4 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001c 00f3 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12
001d 00f2 0000 0052 fffe ff5b 0000 00 00 00 00 ptogdt dd12

>>> We can tell from its location (%70000) that this is a private
>>> arena address in process hptda=91. The 'own' field of the object
>>> record is also hob 91 which again implies an object owned by the
>>> process. That means the object is either a dynamic allocation or
>>> a non-shared segment of the EXE load module, for example it's
>>> stack segment.

```
# .mo 91
hob va flgs own hmte sown,cnt lt st xf
0091 %7b9c484c 8000 ffc9 ff79 0000 00 00 00 00 ptda 0003 c:pmshell.exe
```

>>> This tells us the owner is a PTDA (that is, a process private arena)
>>> and the PID is 3, which is executing PMSHELL.EXE

>>> Note: the PID and executable have been extracted from hob 91 and
>>> displayed in the description area of hob 10.

>>> Now look at the hmte for hob 10.

```
# .mo 19f
hob va flgs own hmte sown,cnt lt st xf
019f %fe862d10 8000 ffa6 0000 0000 00 00 00 mte c:pmwin.dll
```

>>> This is the MTE for pmwin.dll.

>>> The 'own' and 'hmte' of hob 10 tell us that hob 10 was allocated in
>>> the private arena of process PID 3 by pmwin.dll as a result of a
>>> direct or indirect call to pmwin from pmshell.

>>> The flags in hob 10 can give us more information on the
>>> characteristics of hob 10

```
4 0 2 c
0100 0000 0000 1100
  ' ' writeable
  '.....user storage
  '......executable
  '............API located
```

>>> The combination writeable + executable should be interpreted as
>>> R/W storage rather than executable storage. Looking at the page
>>> table entry for %7000 in slot 8 (pid 3) will confirm this:

```
# .s8
Current slot number: 0008
# .p8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32
# dp %7000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00070000* 012f3  frame=0120d  0    0  D  A        U  W  P  pageable
%00070000  0120d  frame=0120d  0    0  D  A        U  W  P  pageable
```

>>> Private arena shared data:
>>> ----------------------------------

```
# .moc 192
*har par cpg va flg next prev link hash hob hal
0249 %feb22250 00000010 %00010000 1c9 024a 024b 014e 0000 0192 0000 hptda=0a6
014e %feb20cbe 00000010 %00010000 1d9 014f 008e 0000 0000 0192 0000 hptda=091
hob har hombxt flgs own hmte sown,cnt lt st xf
0019 024a 0000 0038 0193 0193 0000 00 00 00 00 shared c:pmshell.exe
```

>>> Object 192 has two private arena records pointing to it. One
>>> associated with hptda=0a6 and the other with hptda=91. We established
>>> earlier that hptda=91 is pid 3 and was running pmshell.exe

```
# .mo 2a6
hob va flgs own hmte sown,cnt lt st xf
02a6 %7b9c60d0 8000 ffc9 0000 0000 00 00 00 00 ptda 0003 c:pmshell.exe
```

>>> So hptda=2a6 refers to pid 6, which is also running pmshell.exe

>>> Note the use of the 'link' field in har=249 to point to har=14e. The two
>>> arena records are chained in this way to link all arena records that
>>> share a private data object. The object record points to the head of
>>> the chain.

>>> The 'own' and 'hmte' fields both point to object 193. This tells us
>>> that object 193 is a shared segment of the load module whose
>>> handle is 193.

>>> This may be verified as follows...

```
# .mo 193
hob va flgs own hmte sown,cnt lt st xf
```
### Shared Arena, Global Data:

#### mtos e6

<table>
<thead>
<tr>
<th>har</th>
<th>par</th>
<th>cpq</th>
<th>va</th>
<th>flg</th>
<th>next</th>
<th>link</th>
<th>hash</th>
<th>hob</th>
<th>hal</th>
</tr>
</thead>
<tbody>
<tr>
<td>00d9</td>
<td>fe2b</td>
<td>00000000</td>
<td>7a16</td>
<td>0000</td>
<td>379</td>
<td>00d8</td>
<td>00da</td>
<td>0000</td>
<td>000e</td>
</tr>
</tbody>
</table>

hob = har hobnxt flgs own hton sown, cnt lt st xf
00e6 00d9 0000 008c 00e5 00e5 0000 00 00 00 00 shared c:doscall1.dll
hco=0075d pco=fe6804ec hconext=0084c hptda=005c f=16 pid=0019 c:epm.exe
hco=0084c pco=fe680997 hconext=006d e hptda=0046 f=16 pid=0018 c:epm.exe
hco=006d e pco=fe680271 hconext=0060 hptda=0049 f=16 pid=0013 d:ibmavsd.exe
hco=005c pco=fe67ffe6 hconext=0097 hptda=0041 f=16 pid=0012 c:pmdraw.exe
hco=0097 pco=fe67fe70 hconext=003b hptda=0040 f=16 pid=0010 c:cmd.exe
hco=003b pco=fe67f132 hconext=00327 hptda=0038 f=16 pid=000d c:psl.exe
hco=00327 pco=fe67fe6c hconext=0010 hptda=0036 f=16 pid=000c c:dinfo.exe
hco=0010 pco=fe67e83b hconext=002c hptda=0034 f=16 pid=000b c:mrfile32.exe
hco=002c pco=fe67eb29 hconext=001c hptda=0037 f=16 pid=000a c:pmi.exe
hco=001c pco=fe67eb97 hconext=000a2 hptda=002a f=16 pid=0006 c:pmsl.exe
hco=000a2 pco=fe67ea35 hconext=0030 hptda=0025 f=16 pid=0004 c:harderr.exe
hco=0003 pco=fe67e86b hconext=0009 f=16 pid=0003 c:pmsl.exe
hco=0009 pco=fe67e0e8 hconext=0000 hptda=0019 f=16 pid=0002 c:logdaemon.exe

#### We can tell immediately that this is shared arena global data from the

#### presence of hco= in the arena record. The hco is the handle to the

#### context record. These record the hptda of the process that is accessing

#### shared global data. Each of the VMCOs, that’s sharing the same object

#### is chained in a single linked list from the arena record.
>>> The description to the right of each VMCO is derived from the hptda object.

>>> Note: the .MC command formats a VMCO. Under the Dump Formatter the VMCO chain is not run to completion so we must run the chain manually by using the hconext= field as the VMCO chain pointer.

>>> The 'own' and 'hmte' fields being equal indicate that the object is part of DOSCALL1.DLL. We can check out which object in DOSCALL1 using .lmo

#.lmo e5
hmte=00e5 pmte=0f72d4f4 eflags=8495b954 c:\os2\dll\doscalls.dll
obj vsize vbase flags ipagemap cpagemap hob sel
0001 00001354 1a010000 80009025 00000001 00000002 00eb d00e r-x shr alias iopl
0002 0000ced0 1a020000 80002025 00000003 00000000 00ea d017 r-x shr big
0003 000100a0 1a030000 80001025 00000001 00000002 00e9 d01f r-x shr alias
0004 000002ce 1a040000 80001025 00000012 00000000 00e8 d027 r-x shr alias
0005 0000e54f8 1a050000 8000d025 00000013 00000006 00e7 d02e r-x shr alias conf iopl
0006 000002e8 1a060000 80001023 00000019 00000001 00e6 d037 rw- shr alias
0007 00001b40 1a070000 80001003 00000001 00000002 0000 d03f rw- alias

>>> hob e6 is load module object 6 of doscall1.dll. Furthermore it is a read/write object. We can illustrate this by looking at the page table entries for two of the processes that are accessing hob e6.

#.s8
Current slot number: 0008
# .p8 Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008# 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32 dp %1a060000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a060000* 00e1e frame=00e1e 0 0 D u U W P pageable
%1a060000 00e1e frame=00e1e 0 0 D u U W P pageable

#.s1f
Current slot number: 001f
# .p1f
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
001f# 0006 0003 0006 0001 rdy 062f 7b7e4000 7b9c60d0 7b9ab1f8 1eb8 11 PMSHL32 dp %1a060000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a060000* 0093e frame=00e1e 0 0 C A U W P pageable
%1a060000 00e1e frame=00e1e 0 0 C A U W P pageable

>>> As expected the same page frame (00e1e) is being referenced.

>>> Note also: frame e1e of slot 8 is dirty (Dc=D) and unaccessed (Au=u) while in slot 1f it is clean and accessed. This tends to suggest that frame e1e and therefore page %1a060000 was most recently updated by slot 8 and read by slot 1f before the update took place.

>>> Shared Arena, Instance Data:

#.moc 5ef
*hbar par cpg va flg next prev link hash hob hal
01c6 fbe2170e 00000010 %1a890000 139 01c5 01c7 0000 0000 005ef 0000 =0000
hob har hobnxt flags own hmte sown, cnt 1t st xf
05ef 01c6 04e8 0024 050c 0131 0000 00 00 00 00 priv 0019 e:ipfc.exe
04e8 01c6 02ec 0024 04c6 0131 0000 00 00 00 00 priv 0018 c:epm.exe
02ec 01c6 0457 0024 049c 0131 0000 00 00 00 00 priv 0013 d:ibmavsd.exe
0457 01c6 0432 0024 0410 0131 0000 00 00 00 00 priv 0012 c:pmdraw.exe
0432 01c6 03d6 0024 0420 0131 0000 00 00 00 00 priv 0010 c:cmd.exe
03d6 01c6 0390 0024 0380 0131 0000 00 00 00 00 priv 000d c:ipulse.exe
0390 01c6 035c 0024 036c 0131 0000 00 00 00 00 priv 000c c:info.exe
03c8 01c6 03a7 0024 034e 0131 0000 00 00 00 00 priv 000b c:mrfile32.exe
03a7 01c6 02c7 0024 0317 0131 0000 00 00 00 00 priv 000a c:pmshell.exe
02c7 01c6 0112 0024 02a6 0131 0000 00 00 00 00 priv 0006 c:pmshell.exe
0112 01c6 0000 0024 0091 0131 0000 00 00 00 00 priv 0003 c:pmshell.exe

>>> Object 5ef is an example of shared arena instance data. Each instance of the object has its own object record (VMOB), but they all share the same arena record. Each of these VMOBs is chained from the 'hob' field of the arena record via their 'hobnxt' field.

>>> The VMObs appear as private arena objects, but the arena record does not point to a specific hptda, which distinguishes this case as shared instance data.

>>> As would be expected with shared instance data the owners would differ for each instance object. They are in fact the hptda's for each owner.
Finding Who Owns Memory

Having examined various types of arena, object and context record we now turn our attention to a more commonly asked question: "Who owns a particular location of memory?" To answer this we need to explore the match parameter of the .MA command.

.MAM search for arena records that encompass a given address:

```plaintext
# .mam %123456
har     par      cpg        va    flg next prev link hash hob   hal
008c %feb1fc12 00000080 %00110000 169 00f1 0073 0000 0000 008f 0000 hptda=0091
023b %feb2211c 00001000 %000c0000 1e9 0238 0266 0000 0000 029f 0000 hptda=02a6
02fc %feb231b2 00000010 %00120000 169 02fd 02a2 0000 0000 0318 0000 hptda=036c
0306 %feb2328e 00000010 %00120000 169 0309 0321 0000 0000 03ba 0000 hptda=0380
0312 %feb23396 00000010 %00120000 169 0313 0311 0000 0000 03cd 0000 hptda=034e
032f %feb23614 00000080 %00110000 169 034b 02fa 0000 0000 03e4 0000 hptda=0317
0393 %feb23eac 00000010 %00120000 169 0394 038d 0000 0000 0452 0000 hptda=0410
0412 %feb24996 00000010 %00120000 169 0414 0411 0000 0000 04ef 0000 hptda=04c6
0517 %feb26004 00000010 %00120000 169 0519 0516 0000 0000 05f7 0000 hptda=050c
```

>>> Kernel Debugger needs 'A' explicitly if all contexts are to be searched.
>>> This only affects results from searching private arena addresses.
>>> Note: after fix pack 29 for Warp 3.0 and GA 4.0, .mam under the
>>> Dump Formatter behaves correctly. That is, the 'A' parameter is no longer
>>> "hard-wired".

>>> We can also add the 'C' parameter to chain through the related VMOBs
>>> and VMCOs at the same time.
# .mamc %123456
*har     par      cpg        va    flg next prev link hash hob   hal
008c %feb1fc12 00000080 00000000 00000000 00000000 00000000 0000 hptda=0091
hob har hobnxt flgs own hmte sown,cnt lt st xf
008f 008c 0000 422c 0091 01c0 0000 00 00 00 00 00 priv 0003 c:pmshell.exe

*har par cpg va flg next prev link hash hob hal
023b %feb2211c 00001000 0000c000 00000000 029f 00000000 02a6 hptda=02a6
hob har hobnxt flgs own hmte sown,cnt lt st xf
029f 023b 0000 422c 02a6 01d7 0000 00 00 00 00 00 priv 0006 c:pmshell.exe

*har par cpg va flg next prev link hash hob hal
02fc %feb2211c 00000010 00000000 023b 00000000 02a2 hptda=02a2
hob har hobnxt flgs own hmte sown,cnt lt st xf
02a2 02fc 0000 422c 036c 0371 0000 00 00 00 00 00 priv 000c c:dinfo.exe

*har par cpg va flg next prev link hash hob hal
0306 %feb2211c 00000010 00000000 0309 00000000 03ba hptda=03ba
hob har hobnxt flgs own hmte sown,cnt lt st xf
03ba 0306 0000 422c 0380 035c 0000 00 00 00 00 00 priv 000d c:pulse.exe

*har par cpg va flg next prev link hash hob hal
0312 %feb2211c 00000010 00000000 0313 00000000 03cd hptda=03cd
hob har hobnxt flgs own hmte sown,cnt lt st xf
03cd 0312 0000 422c 034e 0354 0000 00 00 00 00 00 priv 000b c:mrfile32.exe

*har par cpg va flg next prev link hash hob hal
0393 %feb2211c 00000010 00000000 0394 00000000 0452 hptda=0452
hob har hobnxt flgs own hmte sown,cnt lt st xf
0452 0393 0000 402c 0410 ff3e 0000 00 00 00 00 00 priv 0012 c:pmdraw.exe

*har par cpg va flg next prev link hash hob hal
0412 %feb2211c 00000010 00000000 0414 00000000 04ef hptda=04ef
hob har hobnxt flgs own hmte sown,cnt lt st xf
04ef 0412 0000 422c 04c6 04f3 0000 00 00 00 00 00 priv 0018 c:epm.exe

>>> .MAMC is such a frequently used command that it is made the default
>>> specification for .M
>>> Further more, .M will take the default CS:EIP as the match
>>> Suppose we wish to find out what code is being currently executed in
>>> slot 39...

# .s 39
Current slot number: 0039
# .p #
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0039 0019 0010 0019 0001 rdy 061f 7b818000 7b9ca230 7b9adea8 1fc 12 IPPC
# .r
eax=00000000 ebx=00307d90 ecx=00320000 edx=00000000 esi=00001000 edi=00001000
  eip=1a022240 esp=0006d0b4 ebp=00001000 iopl=2 -- -- nv up ei pl nz na pe nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001d6000
005b:1a022240 83c418 add esp,+18
# ln
No Symbols Found
# .m

*har par cpg va flg next prev link hash hob hal
00dd %feb23396 00000010 00000000 00000000 00000000 00000000 0000 hptda=0091
hob har hobnxt flgs own hmte sown,cnt lt st xf
0000 0000 008c 0000 0000 006e 0000 0000 0000 00 priv 0003 c:pmshell.exe

>>> The current cs:eip for slot 39 is executing in doscall1.dll and
>>> has been called either directly or indirectly by ipfc.exe
Finally in this section we answer, "What is the hptda given the PTDA address?"

This required the use of the match parameter with .MO.

.MOM is more restrictive and .MAM. It will only return a result if the supplied address is a precise match for the beginning of a pseudo-object. Since the PTDA is a pseudo-object we can use its address with .MOM:

```
# .p 2a
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
002a  0006 0003 0006 000c blk 021f 7b7fa000 7b9c60d0 7b9ac4e0 1eac 11 PMSHL32
# .mom %7b9c60d0
hob va flgs own hmte sown,cnt lt st xf
02a6  %7b9c60d0  8000 ffcb 0000 0000 00 00 00 ptda 0006 c:pmshell.exe
```

```
>>> The hptda for Pid 6 is therefore 2a6.
```

How to Correlate Named Memory With its Address

Here's how to locate named shared memory, answer who's sharing it and whether a particular address in the shared arena is named.

Named memory is managed using a RMP (Record Management Package). This is a generalised kernel facility for managing global data of variable length. The RMP facility provides allocation, deletion, add and find services. Each RMP user references his RMP structure using a handle. The RMP itself is limited to a maximum of 64K.

```
>>> First locate the handle of named shared memory's RMP:
  #dw sharermpstruc 12
  0400:00004506  0004 0178
     |          Selector for RMP segment
     |          Flags xxxxxxxx
       ......1 = Segment busy
       ......1. = Somebody's waiting
       ......1.. = Segment allocated

>>> The RMP handle is used as the blockid (by ProcBlock) for serialising
>>> RMP manipulations.

>>> Now display the named memory RMP segment:
  #db 178:0
  0178:00000000 00 06 a2 03 5e 02 5e 02-03 00 00 00 00 00 00 00 .".~".
  0178:00000010 83 ff 00 00 11 00 00 00 9b 06 01 00 44 4f 53 5c .DOS
  0178:00000020 43 44 49 42 00 13 00 88-01 47 bd 04 00 50 4d 4c CDIB.
  0178:00000030 52 41 47 2e 4d 54 00 08 00 9f 00 88 01 01 00 RAG.MEM.
  0178:00000040 14 00 91 01 d7 bc 02 00-45 45 42 5e 02 5e 02 5e TLE..
  0178:00000050 4c 43 44 49 42 00 13 00 88-01 47 bd 04 00 50 4d 4c CDIB.
  0178:00000060 43 44 49 42 00 13 00 88-01 47 bd 04 00 50 4d 4c CDIB.
  0178:00000070 9f 00 91 01 00 12 00-a8 01 8f bc 01 00 42 56 ........BVS
  0178:00000080 53 5c 42 56 53 30 31 00-08 00 0f 00 a8 01 01 00 8VS1.
  0178:00000090 12 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .
  0178:000000a0 33 00 08 00 9f 00 9f 00 ab 01-01 00 18 00 03 00 00 .
  0178:000000b0 00 00 53 4d 47 4c 50 50-48 44 45 45 52 2e 44 41 .SMG\PMHDER.R
  0178:000000c0 54 00 08 00 af 01 00 00 00 00 00 00 00 .
  0178:000000d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .
  0178:000000e0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .
  0178:000000f0 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .
```

```
The first 20 bytes form the RMP header, the remainder is a series of
variable length records.

Examining the header first we have:

+00 00 06 = total size of segment (0600)
+02 a2 03 = amount of free space (03a2)
+04 5e 02 = link to first free block (025e)
+06 5e 02 = start of last free block (025e)
+08 03 00 00 00 = heap handle (0003 is kernel heap handle from which
RMP is alloc'd)
+0c 00 00 00 00 = PG alloc/realloc flags
+10 83 ff = hobowner (handle of user of this RMP is ff83)
+12 00 00 = hobmte (hmte of user of this RMP. It's the kernel so 0000)

Check out the owner of this RMP

'mshrmp' is named shared memory management

Records follow the header. They are prefixed by a word length that includes
2 bytes for the length field itself. If the record is free then the high
order bit of the length is set. The data within the record is private to
the owner.

The first record in this RMP is:

...... length 0011

The second record in this RMP is:

Named shared memory management uses two forms of RMP record:
Global - to keep the name, handle, selector and total reference count
Local  - One for each process sharing the named memory. Contains
hptda, hob and ref count for within the given process.

Breaking down record 2 we have:

hob, har, hobnxt, flgs own, hmte own, cnt, lst, st, xf

Named hco=003c8 pco=fe85d503 hconext=003c8 hptda=02fa f=16 pid=0013 c:em.exe
hco=003c8 pco=fe85e303 hconext=00246 hptda=0356 f=16 pid=0009 c:mrfile32.exe
>>> We see 4 owners in accordance with the reference count
>>> note: the owner of the object is 'mshare'
>>> Check out the selector in record 2:

##dl bd47
bd47 Data Bas=17a80000 Lim=00000067 DPL=3 P RW A

>>> In this case it's within the compatibility region so could have used the
>>> CRMA to get %17a80000 directly

>>> The processes sharing the named storage may be obtained directly from local
>>> records in the RMP. A local record is of the following form:

>>> +00 word - length of record (always 0008)
>>> +02 word - hptda of user
>>> +04 word - handle of shared memory object
>>> +06 word - reference count for this ptda.

>>> Scanning through the RMP (for object 0188) we find the following local
>>> records:

>>> 0178:00000030 .. .. .. .. .. .. .. ..-08 00 9f 00 88 01 01 00
>>> 0178:00000100 .. .. .. .. 08 00 58 02-88 01 01 00 ........
>>> 0178:00000120 .. .. .. .. .. .. .. ..-.. .. .. 08 00 fa 02 88
>>> 0178:00000130 01 01 00 .. .. .. .. ..-.. .. .. .. .. .. .. ..
>>> 0178:00000180 .. .. .. .. .. .. .. ..-.. .. .. 08 00 56 03 88
>>> 0178:00000190 01 01 00 .. .. .. .. ..-.. .. .. .. .. .. .. ..

>>> This confirms what was shown in the .mo 188 display, but we have in addition
>>> the reference count for each process.

>>> Finally, we can cut the cake a different way by asking what is all the
>>> named storage being referenced by a particular process. For example
>>> EPM.

>>> Start by finding its slot nos.

- Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD     pPTDA    pTCB     Disp  SG  Name
  0027  0013  0002  0013  0001  blk  0200  7b974000  7bb460d0  7bb2bf24  1e00  12  epm
  0020  0013  0002  0013  0002  blk  0200  7b966000  7bb460d0  7bb2b338  1e00  12  epm

>>> EPM's pPTDA is %7bb460d0. Now find the hptda of this PTDA

##.mom %7bb460d0
hob va flgs own hmte sown,cnt lt st xf
02fa %7bb460d0  8000 ffcb 035b 0000 00 00 00 00 ptda 0013 c:epm.exe

>>> Answer: 2fa.

>>> Now look through the RMP for local records that begin:

>>> 08 00 fa 02

>>> 0178:00000120 .. .. .. .. .. .. .. ..-.. .. .. 08 00 fa 02 88
>>> 0178:00000130 01 01 00 .. .. .. .. ..-.. .. .. .. .. .. .. ..
>>> 0178:00000140 .. .. .. .. .. .. .. ..-.. .. .. 08 00 fa 02 13
>>> 0178:00000150 04 01 00 .. .. .. .. ..-.. .. .. .. .. .. .. ..
>>> 0178:00000160 .. .. 08 00 fa 02 15-04 01 00 .. .. .. ..
>>> 0178:00000180 .. .. 08 00 fa 02 03-04 01 00 .. .. .. ..
>>> 0178:000001a0 .. .. .. .. .. .. .. ..-.. .. .. 08 00
>>> 0178:000001b0 fa 02 f6 02 01 00 .. .. .. .. .. .. .. ..

>>> So, 5 named objects, with hobs=0188, 0413, 0415, 0403 and 02f6
How Memory Aliasing Works

Aliasing is a facility in virtual memory management whereby one or more pages of a memory object may be referenced from an alternative virtual address, possibly from a different process or arena and possibly with different read/write/execute characteristics. It is used extensively by device drivers debugging applications and VDMs.

This example shows how aliasing is represented in the system for a debugging application and how shared storage becomes privatized. There are many ways of creating aliases. The application in this example is IPMD, which uses DosDebug function MapWRAlias to alias the debugee's storage and DosCreateCSAlias to map a code selector to one of his own data segments.

We introduce the memory alias record (VMAL) and the .ML command.

For reference list the thread slots in the system...

```
-p
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD  pPTDA  pTCB  Disp  SG  Name
```
>>> Now list all the busy alias records:

##.ml
hal=0001 pal=%ffe61020 har=00b8 hptda=009f pgoff=00000 f=001
hal=0002 pal=%ffe61028 har=00b9 hptda=009f pgoff=00000 f=001
hal=0003 pal=%ffe61030 har=0018 hptda=009f pgoff=00000 f=001
hal=0004 pal=%ffe61038 har=0019 hptda=009f pgoff=00000 f=001
hal=0005 pal=%ffe61040 har=00b8 hptda=009f pgoff=00000 f=001
hal=0006 pal=%ffe61048 har=00b9 hptda=009f pgoff=00000 f=001
hal=0007 pal=%ffe61050 har=00ba hptda=009f pgoff=00000 f=001
hal=0008 pal=%ffe61058 har=00ba hptda=009f pgoff=00000 f=001
hal=0009 pal=%ffe61060 har=00e7 hptda=009f pgoff=00000 f=001
hal=000a pal=%ffe61068 har=0208 cs=0056 ds=0446 cref=001 f=13
hal=000b pal=%ffe61070 har=020b cs=0056 ds=0446 cref=001 f=13
hal=000c pal=%ffe61078 har=026f cs=007e ds=0446 cref=001 f=13
hal=000d pal=%ffe61080 har=02bf cs=00ae ds=0446 cref=001 f=13
hal=000e pal=%ffe61088 har=02df cs=01ae ds=0077 cref=001 f=13
hal=000f pal=%ffe61090 har=0052 hptda=009f pgoff=00000 f=049
hal=0010 pal=%ffe61098 har=0052 hptda=009f pgoff=00000 f=049
hal=0011 pal=%ffe610a0 har=01b8 hptda=009f pgoff=00000 f=021
hal=0012 pal=%ffe610a8 har=0232 cs=0056 ds=0446 cref=001 f=13
hal=0013 pal=%ffe610b0 har=032e cs=007e ds=0446 cref=001 f=13

>>> hal f & 10 have f=049 = 0000 0100 1001

>>> har=305 is an alias for a linear address in hptda=389

>>> similarly har=306 is an alias for a linear address in hptda=389

>>> Now look closer at hal=f.

##.mac 305

*har par cpg va flg next prev link hash hob hal
0305 %f6e0227 00000010 %00520000 169 0307 0304 00b5 0000 00c1 000f hptda=031a
hal=000e pal=%ffe61090 har=0305 hptda=0389 pgoff=00000 f=049
hal par cpg va flg next prev link hash hob hal
00b5 %f6c6f98 00000010 %1a030000 3d9 00b4 00b6 0000 0000 0000 0000 0000 hco=00502
hob=000e pal=%ffe61090 har=0000 hptda=009f pgoff=00000 f=049
hco=00502 pco=fe85e925 hconext=00473 hpta=003cb f=1c pid=0000 c:epm.exe
hco=00503 pco=fe85e65a hconext=00455 hpta=0019 f=1c pid=0000 c:cmd.exe
hco=00505 pco=fe85e5d4 hconext=00283 hpta=001a f=1c pid=0000 d:ipmd.exe
hco=00506 pco=fe85ecaa hconext=0014a hpta=0266 f=1c pid=0008 c:mrfile32.exe
hco=00507 pco=fe85d68d hconext=00133 hpta=0257 f=1c pid=0005 c:pmshell.exe
hco=00508 pco=fe85db1a hconext=00092 hpta=0248 f=1c pid=0004 c:cmd.exe
hco=00509 pco=fe85d2f5 hconext=00020 hpta=01ae f=1c pid=0003 c:harderr.exe
hco=0050a pco=fe85d0bb hconext=00000 hpta=009f f=1c pid=0002 c:pmshell.exe

>>> Ignoring hco=455 for the moment. har=b5 represents linear address range

>>> %1a030000 in the shared arena. This is in fact a shared object (hob=1c)

>>> which is being accessed by 9 different processes. The hco chain lists those

>>> processes that access this object. hob=1c is one of the objects in doscall1.dll

>>> Let's verify that %al030000 is indeed that same data in each of the contexts.

>>> hco=133 is for pid=4, slot=1c, cmd.exe(1)

##.s 1c

>>> chaining doesn't always work so..

##.mac 305
In each of these cases looked so far, linear address %1a030000 is mapped to the same real address %%a22000.

Now examine the hco flags:

- 9c = 1001 1100
  - | | ||.... User
  - | | |..... Executable
  - | |....... Read
  - |.......... Privatized

Now turn our attention to har=305, hal=f, address %520000 and hco=455. hco=455 has the additional Privatized flag set.

This is no-longer the same storage as in the other contexts. After DPMLINES, was loaded, IPMD created an alias to object 1c reference by DMPLINES.

If we hadn't started with the alias record we could have done a .ml now and looked for the records which referenced hptda=389. As it happens we know already that har=305 is an alias of har=b5. We can check this out by looking at the page tables for %520000 in hptda=031a (pid=b, slot=2c)

%520000 is %%bda000 which is the same real address as %1a030000 in slot=2d. We had to page in %520000 so we should check %1a030000 in slot=2d again, in case it was discarded.

While we are at it, lets check %1a030000 in slot=2c (IPMD)
>> ... and yes as expected IPMD is referencing the shared %la030000, in
>> fact he is referencing both copies.

>> Now let's look at some of the other aliases set up by IPMD
###.mlc 10

*har par cpg va flg next prev link hash hob hal
0306 %fed0228e 00000010 %00540000 169 0308 0307 02ee 0000 0386 0010 hptda=031a
hal=0011 pal=%ffe61098 har=0306 hptda=0389 pgoff=00000 f=049
har par cpg va flg next prev link hash hob hal
02ee %fed0207e 00000100 %00010000 1d9 02ed 02f0 0000 0000 0386 ffff hptda=0389

>> Note: hal=ffff for har=2ee. This is a special hal to indicate a
>> privatized arena - there isn't a context record to put the privatized
>> flag in as this was private arena, shared data.

>> Check out the hptda pids as we have forgotten who 31a and 389 are..
###.mo 31a
hob va flgs own hmte sown,cnt lt st xf
031a %7bb468fc 8000 ffcb 02db 0000 00 00 00 00 ptda 000b d:ipmd.exe
###.mo 389
hob va flgs own hmte sown,cnt lt st xf
0389 %7bb47128 8000 ffe0 0000 00 00 00 00 ptda 000c d:dpmlines.exe

>> Now check the page tables in each processes to prove we are looking
>> at the same thing...
###.ss 2b
###dp %540000 l1
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 vp id=015f0 0 0 c u U W n pageable
###.i %540000
###dp %540000 l1
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 005d2 frame=005d2 0 0 c u U W P pageable
###.ss 2d
###dp %100000 l2
###.i %100000
###dp %100000 l2
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00010000* 02b3 frame=002b3 2 0 D A U W P resident
%00010000 005d2 frame=005d2 0 0 c u U r P pageable
%00011000 vp id=015f1 0 0 c u U r n pageable
###.ss 2b
###dp %540000 l1
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 005d2 frame=005d2 0 0 c u U W P pageable

>> Finally look alias record hal=e. This is a CS Alias of a data
>> segment within in the same process. The hal flags indicate:

>> 13=0001 0011
>> ... Busy (in use)
>> .... CS Alias
>> ........ DS selector valid

###.mlc e

*har par cpg va flg next prev link hash hob hal
02df %fed01f34 00000010 %00350000 1c9 02e2 02de 029d 0000 031b 000e hptda=031a
hal=000e pal=%ffe61088 har=02df cs=01ae ds=0077 cref=001 f=13
har par cpg va flg next prev link hash hob hal
029d %fed01998 00000010 %000e0000 179 02b9 0283 0000 0000 031b 0000 hptda=031a
hal hob hurn flgs own hmte sown,cnt lt st xf
031b 02df 0000 102c 031a 031e 0000 00 00 00 00 priv 000b d:ipmd.exe

>> Check out the page tables...
###dp %350000 12
###.i %350000
###dp %350000 12
### Exploring 32-bit Presentation Manager Under WARP

In this section we look specifically at the messaging function within Presentation Manager (PM).

Sending and receiving messages lies at the heart of how PM applications communicate with each other and the system. Messages may be sent synchronously and posted asynchronously. The mismanagement of messages by an application leads frequently to the 'Bad Application' Pop-up dialog. In extreme cases deadlocks result.

This topic applies to OS/2 V3, which introduced the 32-bit version of Presentation Manager. The previous 16-bit environment has analogous concepts which are briefly explored through a final worked example.

Pre-requisites to any PM debugging requires the following:

- Availability of the symbol file (`PMMERGE.SYM`) for `PMMERGE.DLL`. This should be installed in the same directory as `PMMERGE.DLL` when using the Kernel Debugger or for dump analysis, in the same directory as the Dump Formatter.

- Availability of the PM programming reference from the Programmer’s ToolKit.

- Also of use, is ready access to the C header files from the Programmer’s Toolkit.

We start by giving an overview of the PM messaging environment in which an application’s PM thread operates.

### The PM Messaging Environment

First consider the non-PM application programming model as shown in the following diagram:
Non-PM Application Program Model
This diagram illustrates the following points:

- Non-PM application threads run in a relatively unconstrained environment (compare this with the following situation).
- The Operating System provides a black-box set of services and interfaces.
- The Hardware is not directly accessible by the application.

For PM message threads, the environment is radically different. The key difference is that application code that runs on a PM message thread is effectively a subroutine of the WinGetMsg API even though WinGetMsg is called by the application. The terminology often used to describe this reversal is Program Inversion. WinGetMsg is said to be inverted with respect to the application's message thread.

We see this illustrated in the following diagram.
PM Application Program Model

PM message thread

WinGetMsg

User code

Non-PM application threads

Async for < 0.1 s

OS/2

Hardware

Msg Thread Logic

Wait for message

msg for us?

Yes

User code

No

Post Other Msg Thread
Also illustrated by this diagram are the following points:

- PM Message threads act in a co-operative way. They wait for messages, and pass them on to the appropriate application if not for themselves.
- PM Message threads should spend most of their elapsed time waiting for notification of messages - because of their co-operative nature.
- Application code that runs on the message thread should be limited to very short duration processing. We often speak of the tenth-of-a-second rule, which is intended to imply the transient nature of application code processing rather than a precise measure.
- If a message thread communicates with another thread or the operating system, then this should be done either asynchronously or so as not to violate the tenth-of-a-second rule.

PM Message Queues

PM messages are generated either as the result of user interaction with the system or by the use of various PM APIs. Both PM and non-PM applications may generate PM messages.

Messages may flow:

- synchronously, that is, require processing by the recipient before the sender can continue, or
- asynchronously, that is, where no reposonse is required.

They may flow between threads (inter-thread messages) or from a thread to itself (intra-thread messages).

These characteristics require a message queuing mechanism to be implemented so that message order may be preserved.

Note:

A message's meaning may often depend on the outcome of a preceding message. For example, consider the action of the F4 key after the Alt key has been pressed.

Each PM message thread has two queues or more strictly speaking a message queue and a message list. There may be only one instance of these two structures per thread.

- The message queue is a circular array, the size of which is specified or defaulted by the application when it creates the queue using WinCreateMsgQueue. This queue is used for the receipt of asynchronous messages generated by use of the WinPostMsg API.
- The message list has no depth and is created implicitly by WinCreateMsgQueue. This is used for the receipt of synchronous messages sent using WinSendMsg.

WinCreateMsgQueue also creates a message event semaphore that is posted whenever a message, synchronous or asynchronous, is posted; or a message response is generated. This is the semaphore on which WinGetMsg waits for message notification.

There is a system queue, which is also a circular array. Messages are enqueued on the system queue by the PMDD.SYS device driver as the result of external events deriving directly from:

- Mouse activity
- Keyboard activity
- Use of a light pen
- Timer ticks.

PM maintains knowledge of who the current, mouse, keyboard, pen, and event receivers are. When an external event causes a message to be queued on the system queue, PM posts the message event semaphore of the current receiver of that particular event.
An application may define *Window Procedures* - entry points within the message thread. These are associated with a PM Window and a message queue. They receive control when a message is dispatched, that is dequeued from the message queue or message list. More than one window procedure may be serviced by the same message queue/list. Which one should be dispatched is determined from the HWND, which is one of the parameters associated with a message.

When `WinGetMsg` receives a message event notification, it first checks for the presence of received synchronous messages, if there are any dispatches them directly. Next it looks for an application generated (posted) message and finally for a system queue message.

The application thread explicitly dispatches asynchronous messages using `WinDispatchMsg`.

The System Queue entries are SQMSG structures.

The Application Queue entries are QMSG structures.

The Application Send Message List comprises a chain of SMS structures.

This scenario is illustrated in the following diagram:
An Application Thread's Messaging Structures

The previous section introduced the notion of a message queue and list, of where there is one pair per PM message thread. We now look at these in more detail, with the associated PM system structures that comprise the applications's messaging environment.

**The Message Queue Header (MQ):**

This structure acts as an anchor for all the message processing structures of an application message thread. It is created by `WinCreateMsgQueue`, and the returned `HMQ` (message queue handle) is the address of this structure. PM often refers to the address of a `MQ` as its `PMQ`.

The principle fields of interest are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x14</td>
<td>The current read position of the message queue.</td>
</tr>
<tr>
<td></td>
<td>Since the message queue is a circular array, four pointers have to be maintained: the current read position, current write position, top and bottom of array.</td>
</tr>
<tr>
<td></td>
<td>Each queue entry is a <code>QMSG</code> structure.</td>
</tr>
<tr>
<td></td>
<td>Entries are added to the queue in increasing address order, until the maximum (bottom) is reached then entries are added from the top.</td>
</tr>
<tr>
<td></td>
<td>Removal of entries only involves updating the current read pointer. Thus, a small trace of past message activity may be seen by scanning backwards from the current read pointer to the current write pointer.</td>
</tr>
<tr>
<td>+0x18</td>
<td>The current write pointer.</td>
</tr>
<tr>
<td>+0x24</td>
<td>The <code>Pid</code> of the message thread to which this <code>MQ</code> belongs.</td>
</tr>
<tr>
<td>+0x28</td>
<td>The <code>Tid</code> of the message thread to which this <code>MQ</code> belongs.</td>
</tr>
<tr>
<td>+0x44</td>
<td>The most recent <code>SMS</code> on which a response is awaited.</td>
</tr>
<tr>
<td></td>
<td>The presence of a non-zero value in this field implies that the message thread is currently blocked in <code>WinSendMsg</code> waiting for a response.</td>
</tr>
<tr>
<td></td>
<td>If the message thread recurses, for example through the receipt of a synchronous message, then a subsequent <code>WinSendMsg</code> will cause this field to be updated. The previous contents are saved on the stack.</td>
</tr>
<tr>
<td></td>
<td>A non-zero value in this field is of prime interest when diagnosing hangs. It immediately focuses our attention on the recipient of this message.</td>
</tr>
<tr>
<td>+0x48</td>
<td>The current <code>SMS</code> received.</td>
</tr>
<tr>
<td></td>
<td>This field is non-zero when an <code>SMS</code> is removed from the receive list for processing by its associated window procedure.</td>
</tr>
<tr>
<td></td>
<td>When this field is non-zero, it implies that the thread's window procedure has been dispatched to process a received message.</td>
</tr>
<tr>
<td>+09c</td>
<td>The Received message list.</td>
</tr>
<tr>
<td></td>
<td><code>SMSs</code> are chained from this location pending dispatch.</td>
</tr>
<tr>
<td></td>
<td>Upon dispatch the oldest message is removed from the list and pointed to from offset +0x48 of the <code>MQ</code>.</td>
</tr>
<tr>
<td>+0xa4</td>
<td>The <code>thread slot number</code> of the message thread to which this <code>MQ</code> belongs.</td>
</tr>
<tr>
<td></td>
<td>This is very useful for correlating <code>MQs</code> to threads.</td>
</tr>
</tbody>
</table>
The Send Message Structure (SMS)

The SMS is created for synchronous messages and linked to the receive list (MQ+0x9c) when WinSendMsg is called.

The principle fields of interest are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0c</td>
<td>Pointer to the next (more recent) SMS in the receive list.</td>
</tr>
<tr>
<td>+14</td>
<td>Pointer to the MQ to which this SMS has been sent.</td>
</tr>
<tr>
<td>+18</td>
<td>Pointer to the MQ of the thread from which this SMS was sent.</td>
</tr>
<tr>
<td>+24</td>
<td>Pointer to the WND the represents the Window to which this message has been sent.</td>
</tr>
<tr>
<td>+28</td>
<td>The message Id.</td>
</tr>
<tr>
<td>+2c</td>
<td>Message parameter 1.</td>
</tr>
<tr>
<td>+30</td>
<td>Message parameter 2.</td>
</tr>
</tbody>
</table>

Offsets +0x14 and +0x18 are of prime interest in diagnosing hangs. They enable us to locate the recipient of a message, of which a response is pending and therefore the thread which is causing our thread to remain blocked.

The Queue Message Structure (QMSG)

This is the structure used by applications when calling WinDispatchMsg.

The QMSG is also the form of an entry on the application's message queue.

The principle fields of interest are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>The window handle (HWND). This is an index (ignoring the high-order bit) in to the handle table. From the handle table we can obtain the equivalent PWND or pointer to the WND.</td>
</tr>
<tr>
<td>+4</td>
<td>The message Id.</td>
</tr>
<tr>
<td>+8</td>
<td>Message parameter 1.</td>
</tr>
<tr>
<td>+c</td>
<td>Message parameter 2.</td>
</tr>
</tbody>
</table>

The Handle Table

This is a global table that is used to correlate window handles (HWNDs) with pointers to WNDs (PWNDs). The table comprises a 0x20 byte header with 8-byte entries. The first word of each entry is a PWND and the second a Boolean flag, which if non-zero, indicates that an HWND/PWND combination is non-deleteable.

The handle table may be located from the address at symbol:

pHandleTable

The Window Structure (WND)

This structure represents a window. It is created by WinCreateWindow.

The WND has two main functions:

>> It acts as the link between the MQ and the thread's window procedure
>> It establishes the WND hierarchy.

The principle fields of interest are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>Next sibling WND.</td>
</tr>
<tr>
<td>+4</td>
<td>Parent WND.</td>
</tr>
<tr>
<td>+8</td>
<td>First Child WND.</td>
</tr>
</tbody>
</table>
+28  The pointer to the MQ that will queue messages sent and posted to this window.

+2c  The window's HWND.

+30  A Boolean, which if non-zero, indicates that the window procedure address is a 16-bit far pointer.

+34  The address of the window procedure.

This scenario is illustrated in the following diagram:
PM Application Message Processing Overview

Posted Msg Q

MQ

+14

+18

+16

SMS

Current SMS

SMS

Seat SMS

Received Msg List

HandleTable

SMS

SMS

SMS

+24

+40

+36

Window Procedure
PM Message Processing Logic

The following sections provide a summary of the essential internal logic of PM's message handling. This is provided to give the reader sufficient understanding that will enable most application problems, especially those that cause hangs, to be identified. In most cases hangs in the PM environment are caused by a misuse or misunderstanding of the message thread model, especially the way in which message threads act in a co-operative manner.

An outline is given for each of the following:

- **WinGetMsg** logic.
- **WinSendMsg** logic.
- Waiting for Message Activity.
- **WinSetFocus** logic.

**Note:**

In each of these outlines, the added complication of calling hooks has been omitted.

--------------------------------------------

WinGetMsg Logic

*WinGetMsg* operates essentially as a loop that waits for message activity and returns messages to the user. Conceptually the application's code acts as an inverted subroutine of *WinGetMsg*. This was illustrated in *The PM Messaging Environment*.

These are the essential steps in *WinGetMsg* processing:

- When *WinGetMsg* wakes, it first unlocks the System Queue if owned or if it is the *active thread*. The *MQ* of the current system queue owner is pointed to by *pmqsyslock*. This is set to zero if this points to the *MQ* of the current thread.

  The *active thread* is the thread that has the right to unlock to system queue if locked by another thread. Normally this is the thread that manages the *MQ* of the window in focus. Normally the thread that has locked the system queue is the *active thread*.

- The received list is checked.

  If SMSs are queued then each is removed successively and the corresponding window procedure is called.

  Received messages, that is messages sent via *WinSendMsg* to this thread, are not returned by *WinGetMsg*. The window procedure is called directly.

- The application's queue is checked for posted messages.

  If one is found it is dequeued and returned to the application.

- If no posted message is found then *WinGetMsg* tries to process the system queue.

  We attempt to lock the system queue if free.

  if *pmqsyslock* is zero it is set to the current thread's *MQ* address.

  If the lock was successful then we peek the next system queue message.

  If the lock was unsuccessful we return to the beginning of the loop and wait on the message queue semaphore.
• If the next system message is for this thread then it is dequeued and returned to the application with the system queue lock still held.

The possibility of holding the system queue lock while running in user code is vital to note. While this happens only active thread can dequeue a system queue message. The reason for holding the system queue lock is for performance. It is likely that one system message will be followed by a sequence for the same thread. If the lock was released, unnecessary processing on other message queue threads would take place. More recently queued messages could not be processed out of turn anyway, since the interpretation of a system message depends upon the outcome of the preceding system message.

• If the next system message is for another application then the system queue is unlocked.

• The other application is made the current input receiver.

PM distinguishes between current mouse, keyboard and event receiver. WinGetMsg makes the other application the current mouse, keyboard or event receiver depending upon the message category.

• Finally the other application’s message queue semaphore is posted. WinGetMsg returns to the beginning of its message loop by waiting on its own message semaphore for more message activity.

This processing is illustrated in the following diagram:
WinSendMsg Logic

These are the essential steps in WinSendMsg processing:

- We check to see if the message is being sent to the same thread.
  WinSendMsg to the same thread is known as an *intra-thread* send.
  WinSendMsg to another thread is known as an *inter-thread* send.

- If intra-thread then the message is dispatched immediately, from within WinSendMsg.
  This behaviour implies that a window procedure may recurse many times, even if waiting for a response to a WinSendMsg.

- If inter-thread then the message is enqueued to the receive list of the recipient’s MQ.

- Active thread status is transferred to the receiving thread (if currently owned).
  This allows the receiver to unlock the system queue if it had been locked by the current thread *and* the current thread is the active thread.

- The receiver's message queue semaphore is posted.

- WinSendMsg waits on the current thread's message queue for a response to the sent message.

This processing is illustrated in the following diagram:
WinSendMsg Essential Processing

WinSendMsg

Same Thread?

Yes

No

Has Intra-thread send?

Req Msg on Recv's EMT3 Queue

Transfer Active Thread to Recv

Request Recv's MsgQ Semaphore

Wait on MsgQ Sem

Send Recv Msg

Make Sender Active Thread

Post Sender's MsgQ semaphore

Return to App.

Sender's Thread Context

Receiver's Thread Context

Send Recv Proc

Wait Proc
Waiting for Message Activity

In order to process synchronous messages as swiftly as possible, PM always checks the receive list of the current thread for pending SMSs before waiting on an internal PM semaphore.

If SMSs are found queued, they are successively dispatched.

Only when the receive list has been processed does PM finally wait on a semaphore.

This applies particularly to the message processing semaphore, but also equally to semaphores that serialise access to resources such as the .INI files.

WinSetFocus Logic

WinSetFocus has a subtle bearing on message processing since it selects a new active thread and new current input receivers for system messages.

Note:

Focus may be changed by a third party.

These are the essential steps in WinSetFocus processing:

- WM_FOCUSBANCE is sent to the message thread losing the focus.
- The current window in focus is changed.
  - pwndfocus points to the WND of the focus owner.
- Unlock the system queue if locked.
  - The MQ of the current system queue owner is pointed to by pmqsystock. This is set to zero if it points to the MQ of the current focus owner.
- The target window’s message thread is marked as the new focus owner.
  - pmqfocus is set to the address of the new focus owner’s MQ.
- The target thread’s message queue is made the current mouse and keyboard input receiver.
  - pmqMouseWake and pmqKeyWake are set to the address of the new focus owner’s MQ.
- The new focus owner’s message thread is marked as the new active thread.
- A priority boost is applied to the new focus owner’s message thread.
- WM_FOCUSBANCE is sent to the message thread gaining the focus.

This processing is illustrated in the following diagram:
Application Not Responding to Messages Logic

The Application Not Responding to Messages dialog, or BadApp dialog as it is sometimes referred to, appears after Ctrl-Esc has been hit and the system has not been able to display the task list.

The essential logic for this processing is as follows:

- Ctrl-Esc is hit and 5 second timer is started.
- If the task list appears before the time-out then all is well, if not we check for who might be holding things up.
- We try for 5 seconds to obtain the User PM Semaphore. If unsuccessful then the Pid, Tid and Session Id of the owner is saved in the QHPSTRUCT.
- If pmqsyslock is owned then the Pid, Tid and Session Id of the owner is saved in the QHPSTRUCT.
- If pmqfocus is owned then the Pid, Tid and Session Id of the owner is saved in the QHPSTRUCT.
- We now enter a second wait of a further 3 seconds, after which, we build a second QHPSTRUCT.
- If the Tid and Pid are different and we have not yet had an acknowledgement from the task list then we wait a further 12 seconds, on the assumption the processing is slow, but not hung.
- If the Tid and Pid are the same or we have expired on our third time-out then we set fBadAppDialog true and reset the cause for the hang:
  - If User_Sem held then release User_Sem
  - if system queue locked then reset pmqsyslock
  - if focus owner hung, then reset pmqfocus

Report the hanging application in the BadApp dialog.

This processing is illustrated in the following two diagrams:

BadApp Dialog Logic
BadApp Dialog Processing

Ctrl-Esc

T/L Request

T/L Ack

Wait

Display Fast-List

Acknowlegde T/L Request

Query Hung Proc

Exit

T/L Ack

Wait

Query Hung Proc

Exit

Display Dialog

Same Fid/Tid

Different Fid/Tid

Exit

T/L Ack

Wait

12s T/O
Query Hung Process Logic
The flags in the QHPSTRUCT indicate the detected reason for hanging. These may be a combination of:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QHP_SYSQUEUELOCK</td>
<td>0x0001</td>
<td>System Queue Locked</td>
</tr>
<tr>
<td>QHP_SENDMSGLOCK</td>
<td>0x0002</td>
<td>Waiting for a response to WinSendMsg</td>
</tr>
<tr>
<td>QHP_CLIPBRDLOCK</td>
<td>0x0004</td>
<td></td>
</tr>
<tr>
<td>QHP_WINDOWLOCKED</td>
<td>0x0008</td>
<td></td>
</tr>
<tr>
<td>QHP_VISRGNLOCKED</td>
<td>0x0010</td>
<td></td>
</tr>
<tr>
<td>QHP_LOCKWINDOWUPDATE</td>
<td>0x0020</td>
<td></td>
</tr>
<tr>
<td>QHP_FSRUSERHANG</td>
<td>0x4000</td>
<td>Waiting for the User_Sem</td>
</tr>
<tr>
<td>QHP_INPUTPROCESSED</td>
<td>0x8000</td>
<td></td>
</tr>
</tbody>
</table>

--------------------------------------------

Useful Symbols for PM Structures

The following list is a small selection of global symbols from PMMERGE.SYM that will be of help in locating the structures associated with message handling:

- **pmsemaphores**
  This is the label for the table of PMSEM and GRESEM semaphore structures. Offset +0x20 is the location of the User Semaphore. If this is owned and not released within the time-out period after Ctrl-Esc has been hit, then the Application Not Responding to Messages reports the semaphore owner as the culprit.

- **pmqSyslock**
  The PMQ of the thread that has locked the system queue.
  This is the first place to look when investigating a hang in a PM application. The system uses this to name a bad application when the Application Not Responding to Messages dialog appears.

- **pmqFocus**
  The PMQ of the window that has the focus.
  if pmqSyslock is zero, the system uses this as a second choice for the bad application when the Application Not Responding to Messages dialog appears.

- **pmqKeyWake**
  The PMQ of the current keyboard event receiver.

- **pmqMouseWake**
  The PMQ of the current mouse event receiver.

- **pmqEventWake**
  The PMQ of the current miscellaneous event receiver.

- **pwndFocus**
  The PWND of the window currently in focus.

- **pmqShutdown**
  The PMQ of the thread sent a WM_QUIT and being waited on to terminate. If Shutdown doesn't complete, then this could be a good place to start investigation of the problem.

- **pmqCapture**
  The PMQ of the thread that has the mouse captured.

- **pwndCapture**
  The PWND of the window associated with mouse capture.
The PWND of the System Modal window.

The PMQ that has visible regions locked.

The PMQ of the thread currently in WinTrackRect.

The PMQ of the thread that has update locked.

The address of the handle table.

The MQ of the system input queue.

The system queue is headed by a partial MQ since it does not require the fields to support a receive list.

The PMQ of the 1st thread of the 1st Shell Process.

This thread is responsible for starting and re-starting the Workplace Shell.

The PMQ of the 1st thread of the 2nd Shell or Workplace Shell Process.

This thread is the main thread of the desktop PM application.

The address of the application anchor block registers (AAB).

AAB registers are allocated for each PM application message thread. This is located in thread local memory, which implies that it is correct only for the current thread context. The Thread Local Memory Area is saved in the TCB and restored when the thread is made current. Since the TCB may be located in System storage under any context then a thread's PMQ may be found in any context from its TCB.

The PWND of the Object Window.

This is the parent or owner of all non-display windows.

The PWND of the Desktop Window.

This is the parent of owner of all displayable windows.

When a thread is blocked at approximately offset +0x155 into SleepPMQ then it is waiting on the message queue semaphore for new messages or responses to outstanding sent messages. Offset +0x30 from the current stack pointer usually contains the PMQ for the current thread.

All MQs are chained on a single linked master list from offset +0x0 of the MQ. The current head of the master list is pointed to by pmqList.

All SMSs are chained on a single linked master list from offset +0x0 of the SMS. The current head of the master list is pointed to by psmsList.

This is the label of the QHPSTRUCT saved the first time a time-out occurs after Ctrl-Esc has been hit and the Application Not Responding to Messages dialog appears.

A Boolean that indicates when the Application Not Responding to Messages dialog has been displayed.

Useful PM Structures
The following diagrams illustrate the main PM structures for the messaging function. These are laid out as if viewed under the Kernel Debugger or Dump Formatter by displaying them using the DD Command.

- PM Message Queue Header (MQ)
- PM Window Structure (WND)
- PM Message Structures (SMS, QMSG, SQMSG)
- PM Application Anchor Block Registers
- Stack Layout at Useful Entry Points

PM Message Queue Header
PM Message Queue Header viewed as double-words

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>Next HQ in water list</td>
</tr>
<tr>
<td>+10</td>
<td>Number queued</td>
</tr>
<tr>
<td>+10</td>
<td>Length</td>
</tr>
<tr>
<td>+10</td>
<td>Queued</td>
</tr>
<tr>
<td>+10</td>
<td>Top of Queue</td>
</tr>
<tr>
<td>+10</td>
<td>Depth</td>
</tr>
<tr>
<td>+10</td>
<td>Queue</td>
</tr>
<tr>
<td>+20</td>
<td>Next QBSS to Read</td>
</tr>
<tr>
<td>+20</td>
<td>Next QBSS to Write</td>
</tr>
<tr>
<td>+20</td>
<td>2ID</td>
</tr>
<tr>
<td>+20</td>
<td>TID</td>
</tr>
<tr>
<td>+20</td>
<td>E2CID</td>
</tr>
<tr>
<td>+30</td>
<td>MSQ Event Set handle</td>
</tr>
<tr>
<td>+40</td>
<td>Current Sent SMS WinSendMsg</td>
</tr>
<tr>
<td>+40</td>
<td>Current Rcvd SMS WinSendMsg</td>
</tr>
<tr>
<td>+93</td>
<td>Rcvd SMS list pending dispatch</td>
</tr>
<tr>
<td>+93</td>
<td>Thread Slot</td>
</tr>
</tbody>
</table>
PM Window Structure (WND)
## PM Window Structure (WND)

*viewed as double-words*

<table>
<thead>
<tr>
<th>Address</th>
<th><strong>NextSibling WND</strong></th>
<th><strong>Parent WND</strong></th>
<th><strong>Child WND</strong></th>
<th><strong>Owner WND</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+10</td>
<td></td>
<td></td>
<td></td>
<td>User specified id</td>
</tr>
<tr>
<td>+20</td>
<td></td>
<td></td>
<td></td>
<td>EQQ</td>
</tr>
<tr>
<td>+30</td>
<td></td>
<td></td>
<td></td>
<td>PQQ</td>
</tr>
<tr>
<td>+40</td>
<td>16-bit BOOL</td>
<td>Window Proc address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+60</td>
<td>Window words start here</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

*Note: The table represents the structure of a window in memory, showing the relationship between parent and child windows, and other relevant data.*
PM Message Structures (SMS, QMSG, SQMSG)
PM Send Message Structure (SMS) viewed as double-words

<table>
<thead>
<tr>
<th>+0</th>
<th>Next SMS in Send list</th>
<th>Send list Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>+13</td>
<td>time</td>
<td>Sender MQ</td>
</tr>
<tr>
<td>+30</td>
<td></td>
<td>Receiver MQ</td>
</tr>
<tr>
<td>+33</td>
<td></td>
<td>Result</td>
</tr>
</tbody>
</table>

PM Queue Message Structure (QMSG) viewed as double-words

<table>
<thead>
<tr>
<th>+0</th>
<th>HCNID</th>
<th>Posted Msg ID</th>
<th>X8</th>
<th>X92</th>
</tr>
</thead>
<tbody>
<tr>
<td>+13</td>
<td>time</td>
<td>X Co-ord</td>
<td>Y Co-ord</td>
<td></td>
</tr>
</tbody>
</table>

PM System Queue Msg Structure (SQMSG) viewed as double-words

<table>
<thead>
<tr>
<th>+0</th>
<th>Msg ID</th>
<th>X8</th>
<th>X92</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PM Application Anchor Block Registers
PM Application Anchor Block

TCB

Thread Local Memory Area

+A+6
TCB + A

AAB Regs

Last Error | DMQ | Error Info
+A+3 | +A+4 | +A+5

paAABRegs (current thread only)

% AAB Regs (copy)
Stack Layout at Useful Entry Points
Stack Frames for Common Entry Points viewed as double-words

**Win32PostMsg Entry Point**

**Win32SendMsg Entry Point**

**Window Procedure Entry Point**

<table>
<thead>
<tr>
<th></th>
<th>Next Frame pointer</th>
<th>Return Address following call to entry point</th>
<th>MSG</th>
<th>MSG ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>WP1</th>
<th>WP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Win32DispatchMsg Entry Point**

<table>
<thead>
<tr>
<th></th>
<th>Next Frame pointer</th>
<th>Return Address following call to entry point</th>
<th>MSG</th>
<th>Pointer to WM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PM Worked Examples Under WARP

We give two examples of diagnosing common application problems:

Example 1 - A trap in PMMERGE.DLL caused by an application fault.

Example 2 - A hang in the WorkPlace, again caused by an application fault.

Further techniques are illustrated in:

How to find the MQ of any thread.
How to find the MQ of a BadApp Application.
Finding Application and System Queue Elements.

Further examples, with annotated solutions, may be found on the accompanying CD-ROM in the TURKEY lab exercise.

Example 1 - A Trap in PMMERGE.DLL

Steps for analysing traps in PM DLLs:

• Intercept the trap at the point of failure.
• Unwind the stack to the application call.
• Validate the parameters to the API call.
• If necessary, determine how the user routine was invoked by examining the MQ and looking for dispatched messages or by unwinding the stack further.

This example is of a trap in PMMERGE.DLL, but caused by an application fault!

Because we have a trap E, we set the fatal vector under the Kernel Debugger (or use TRAPDUMP=ON in CONFIG.SYS to take a dump) then re-create the problem.

```plaintext
##vsf *
%g
Trap 14 (0EH) - Page Fault 0006, Not Present, Write Access, User Mode
eax=00000007 ebx=00273fcb ecx=00000001 edx=00000007 esi=12d3e089 edi=00000000
eip=1bd3d261 esp=00273ebc ebp=00273f1c iopl=2 rflags=0x0 pf -- af cm ds:0028 es:0000 fs:0000 gs:0000
cr0=00000001 cr2=00000000 cr3=00000000 cr4=00000000
 Mavericks.
```

We have trapped in a DLL, probably PMMERGE.DLL, certainly not in the user's .EXE code. We unwind the stack to find the return address in the user's .EXE

```plaintext
##dd %ebp
%00273f1c 00273f5c 1bd3d05b 00000000 00000005
%00273f2c 00080001 00000000 00000013 00010423
%00273f3c 00190008 00000000 00000001 00000014
%00273f4c 00000000 80000144 00000001 00190008
%00273f5c 00273fa8 1bd3893 80000144 00000071
%00273f6c 00280018 00000000 000103ec 00000000
%00273f7c 00000000 00190008 00273f5c 00000000
%00273f8c 00000000 00000000 80000144 12d31630
%00273f9c 00000000 00190008 00273ff4
%00273fac 000103ec 00000000 00000000 00000000
```
Not all the stack was paged in to physical memory, but never mind. Enough is there to allow us to find the return address to the user’s application code.

Following the base pointer (EBP):

The return address is %10932. We inspect the code just before this address.

This call was to a routine at %1bd037b8. LN doesn't give us a useful symbol (our .EXE is being selected instead of the correct .DLL symbol). We find out who owns this address from the memory management control blocks.

We can see that our call was to an entry point in PMMERGE.DLL. We need to activate PMMERGE's symbols.
parameters to WinDispatchMsg. These are:

```
HAB   00190008
PQMSG 00273fcc
```

The QMSG at %273fcc is also in the stack we dumped:

```
%00273fcc  80000144 00000071 00280018 00000000
%00273fdc  0092d87d 0000027a 000000f0 00000000
```

The first parameter is the HWND. We convert this to a PWND, dump the WND and look for the window procedure entry point.

```
##dd phandletable 11
9f3f:0000ab78  12d50000

##dd %12d50000+20+(8*144) 12
%12d50a40  12d31494 00000000

##dd %12d31494
%12d31494  12d31838 12d3c974 00000000 12d3c974
%12d314a4  00c80262 0104029e 80000000 00000008
%12d314b4  12d314f0 00000004 12d31630 80000144
%12d314c4  00000000 00103ec 00000000 00000000
%12d314d4  12d3147c 00000000 00000000 00000000
%12d314e4  00000000 2050534d 00000034 12d31894
%12d314f4  00004b4e 00000000 000fc4e 00000019
%12d31504  00000000 00103ec 00000000 00000000
```

Note:

We could have used the following more complex single command construct to achieve the same result:

```
##dd %(dw(%(dw(phandletable))+20+(8*144)))
%12d31494  12d31838 12d3c974 00000000 12d3c974
%12d314a4  00c80262 0104029e 80000000 00000008
%12d314b4  12d314f0 00000004 12d31630 80000144
%12d314c4  00000000 00103ec 00000000 00000000
%12d314d4  12d3147c 00000000 00000000 00000000
%12d314e4  00000000 2050534d 00000034 12d31894
%12d314f4  00004b4e 00000000 000fc4e 00000019
%12d31504  00000000 00103ec 00000000 00000000
```

The window procedure entry point is at offset +0x34.

We now unassemble this:

```
##u %000103ec
%000103ed  8bec          mov     ebp,esp
%000103ef  83ec08         sub     esp,+08
%000103f2  8b4508         mov     eax,dword ptr [ebp+08]
%000103f5  a32c0d0200     mov     dword ptr [00020d2c],eax
%000103fa  8b450c         mov     eax,dword ptr [ebp+0c]
%000103fd  e93a000000     jmp     %0001043c
%00010402  8be0          mov     eax,eax
%00010404  ff7508         push    dword ptr [ebp+08]
%00010407  b001          mov     al,01
%00010409  e8322dcd21b    call    WIN32QUERYANCHORBLOCK (%1bd03140)
%0001040e  8945fc         mov     dword ptr [ebp-04],eax
##u
%00010411  6a00          push    +00
%00010413  6a14          push    +14
%00010415  6a01          push    +01
%00010417  6a00          push    +00
%00010419  ff75fc         push    dword ptr [ebp-04]
%0001041c  b005          mov     al,05
%0001041e  e81dcd21b      call    WIN32LOADSTRING (%1bd3d040)
%00010423  83c418         add     esp,+18
%00010426  eble          jmp     %00010446
```
We notice that we trapped in an internal routine called LoadStrMsg and that we have called WinLoadString in the window procedure. Could these be related?

We see from the PM Programming Reference that WinLoadString has 5 parameters. The right most is a pointer to a buffer and we see that the window procedure has pushed 0 on the stack this will surely cause WinLoadString to trap at some point. How do we make this supposition less circumstantial and more concrete?

Clearly, for EBP to take us back to a call to WinDispatchMsg, without finding a stack frame from the window procedure implies that PMMERGE is using optimised code when the trap occurred. That is, the conventional use of EBP is not in place - and this does occur in many internal routines in PMMERGE, for performance reasons. If we scan back through the stack we notice the address %10423 occurring shortly before (in time) the call to LoadStrMsg. This address is the return address from the WinLoadString call in the window procedure. It would seem therefore that we have called that API with the bad parameter as suspected!

Example 2 - A Hang in a PM Application

Steps for analysing hangs in PM applications:

- Determine whether there is a general hang in the PM environment, or a just in one application. If the latter then proceed with normal hang analysis.
- Check whether the User_Sem is owned. If it is then this may be an indication of a problem. Determine the owner and their thread status.
- Check pmqsyslock to see if the system queue is locked. If it is, then determine the owner of the lock and their thread status.
- Check pmqfocus if neither of the preceding checks reveals anything informative. Determine the thread in focus and its status.
- If pmqfocus is a shell thread, check fBadAppDialog. If it is non-zero then analyse the QHPSTRUCT at label qhpsbadapp.
- If none of the preceding steps yields any results then check the shell processes. In particular pmqshell and pmqshell2. Most of the time these threads should be waiting for a message to arrive. Any other state should be transient.

This example is of a hang in the WorkPlace caused by a PM application fault.

First we check out whether the User_Sem is held, whether the system queue is locked and if necessary who has the focus.

```plaintext
%0010428 8b4508 mov    eax,dword ptr [ebp+08]
%001042b e8d0fbffff call    main (%00010000)
%0010430 eb14 jmp     %00010446
```
No one Owns the User_Sem since words at offsets +0x8 and +0xa are both zero.

We see that the system queue is held by slot 2e, who happens to be blocked in PMMERGE which is waiting for message activity. We also notice that at MQ+44 there is a non-zero value, which indicates that this thread has called WinSendMsg and is waiting for a response.

We investigate the WinSendMsg by examining the SMS pointed to by MQ+44

```
%12d3ff5c %12d3ff5d %12d3ff6c %12d3ff7c %12d3ff8c %12d3ff9c %12d3ffac %12d3ffbc %12d3ffcc
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ff6c 00000000 00000000 00000000
%12d3ff7c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ff8c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ff9c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ffac 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ffbc 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%12d3ffcc 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

The target MQ for the sent message is at offset +18, i.e. %12d32cb4

We find out who this is (the slot number is at MQ+a4).

```
%ab9c9408 %ab9bc6c0 %ab9c96e8 %ab9bc6c0
00e90000 00054531 0000325 00000000
%ab9c96e8 00000000 00000000 00000000
%ab9bc6c0 00000000 00000000 00000000
```

Offset +a4 gives us the slot number which turns out to be another thread of the turkey application. The status of this thread is crt! This indicates that some other thread in the same process has entered critical section, furthermore slot 28 would be ready to run had it not been for the critical section thread. Clearly this is why our application has hung the PM messaging function. The real culprit is the user of Critical Section, who is it?

The PTDA contains the address of the TCB in critical section. The TCB offset +0 contains the thread id followed by the slot number.

```
%ab9c9408 %ab9bc6c0 %ab9c96e8 %ab9bc6c0
11 11 11
```

Our application has perpetrated one if not two faults:

- First, we are using DosEnterCriticalSection in a PM application. This is a very heavy-handed way of serialising and likely to impact PM message processing, particularly if one of the other threads in the application holds the system queue lock.
- Secondly and more seriously, the thread that has entered critical section has subsequently called an API. The consequences of this are unpredictable and can lead to a hang as illustrated. Furthermore, this would apply whether or not the application was
How to Find the MQ of Any Thread

This example illustrates a basic technique for finding the MQ for a specific thread.

We find the MQ for thread slot 8:
```
# .p8
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD    pPTDA    pTCB    Disp  SG  Name
*0008# 0004 0001 0004 0001  blk  0500 ab596000 ab9c7020 ab988bf0 1ed0 01  pmshell
```
```
%ab988c64 00000000 00070000 00041304 12d2ca34
%ab988c74 12d2ded8 00000000 00000000 00000000
%ab988c84 00000000 00000000 00000000 00000000
%ab988c94 00000000 00000000 00000000 00000000
%ab988ca4 00000000 00000000 00000000 00000000
%ab988cb4 00000000 00000000 00000000 00000000
%ab988cc4 00000000 00000000 00000000 00000000
%ab988cd4 00000000 00000000 00000000 00000000
```

The TCB address is found from the .P output.

Offset +0x74 into the TCB is the saved thread local memory area.

Offset +0x08 into the TLMA are the AAB registers.

The first is the last PM error to occur on this thread. In this case severity 4 error code 1304.

The next double-word is the PMQ.

We can verify this by displaying it and checking the offset +0a4 is the same thread slot number.

Notes:

After fix-pack 7 the TCB in WARP is extended by 4 bytes. The TLMA begins at TCB+0x78.

Since AAB is allocated using DosAllocThreadLocalMemory its location in the TLMA is depended how many TLMA allocations are made prior to PM initialising in the thread. In practice this is normally found at TLMA offset +0x8, but if the AAB is displaced it can usually be located by searching the TLMA for the MQ address which is normally in the range 12000000 to 14000000.

```
#dd %12d2ca34
%12d2ca34 00000000 00000020 00000064 12d2cadc
%12d2ca44 12d2d75c 12d2cd3c 12d2cd3c 80002fff
%12d2ca54 80008000 00000004 00000001 00000001
%12d2ca64 80030038 0032bd01 000000ce 00000050
%12d2ca74 00000001 00000000 00000000 00000010
%12d2ca84 12d29c74 00000000 00000000 00000000
%12d2ca94 00000000 00000000 00000000 00000000
%12d2ca4 00000000 00000000 80000006 00000006
#d
%12d2cbb4 00000000 00005453 0000024f 00000000
%12d2cbb4 12d2c910 0b6f0c02 00000000 00000000
%12d2cbb4 00000001 00000000 80000007 00002f43
%12d2cbb4 00000004 0000128 00000000 00010000
%12d2cbb4 00000000 00000000 80000007 00002f43
%12d2cbb4 00000004 0000128 0001168e 00010000
```

How to Find the MQ of a BadApp Application
This example illustrates how to find the MQ of the application that causes the BadApp dialog to appear.

As discussed in Application Not Responding to Messages Logic pmqsyslock, pmqfocus and the User_Sem PM semaphore owner will be reset when the BadApp dialog is displayed.

To find the MQ of the bad application under these circumstances we look at the Query Hung Process Structure (QHPSTRCUT).

```
##db fbadappdialog l1
9f3f:0000035c 01
##dd pmqsyslock l1
9f3f:0000ed14 00000000
##dd pmqfocus l1
9f3f:0000e0fc 12d2b0f0
##dd %12d2b0f0
%12d2b0f0 12d2b344 00000020 0000000a 12d2b198
%12d2b100 12d2b08c 0bf00002 00000000 00000000
%12d2b110 12d2b08c 00000000 0000000a 00000000
%12d2b120 12d2b08c 00000000 0000000a 00000000
%12d2b130 12d2b08c 00000000 0000000a 00000000
%12d2b140 12d2b08c 00000000 0000000a 00000000
%12d2b150 12d2b08c 00000000 0000000a 00000000
%12d2b160 12d2b08c 00000000 0000000a 00000000
##d
%12d2b170 10ff0000 00005453 0000024f 00000000
%12d2b180 12d2b08c 0bff0002 00000000 00000000
%12d2b190 12d2b08c 00000000 0000000a 00000000
%12d2b1a0 12d2b08c 00000000 0000000a 00000000
%12d2b1b0 12d2b08c 00000000 0000000a 00000000
%12d2b1c0 12d2b08c 00000000 0000000a 00000000
%12d2b1d0 12d2b08c 00000000 0000000a 00000000
%12d2b1e0 12d2b08c 00000000 0000000a 00000000
##p 18
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
0018  0004 0001 0004 000f blk 0500 ab5b6000 ab9c7020 ab98ab70 1ed0 01 pmshell
fbadappdialog
is non-zero, which indicates that the BadApp dialog has been displayed.

pmqsyslock is not owned.

pmqfocus points to a shell thread, in fact the BadApp dialog thread.

So we look at qhpsbadapp
```
##dw qhpsbadapp l4
9f3f:0000e490 0002 000e 0008 0016
##p
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
0001  0001 0000 0000 0001 blk 0100 ffe38000 ffe3aa04 ffe3a80c 1eb4 00 *ager
0002  0001 0000 0000 0002 blk 0200 ab58a000 ffe3aa04 ab988020 1f3c 00 *tsd
0003  0001 0000 0000 0003 blk 0200 ab58c000 ffe3aa04 ab988218 1f50 00 *ctxh
0004  0001 0000 0000 0004 blk 081f ab58e000 ffe3aa04 ab988410 1f48 00 *kdb
0005  0001 0000 0000 0005 blk 0800 ab590000 ffe3aa04 ab988608 1f20 00 *lasyw
0006  0001 0000 0000 0006 blk 0800 ab592000 ffe3aa04 ab988800 1f3c 00 *asyncr
*0007# 0004 0001 0004 0001 blk 0500 ab596000 ab9c7020 ab988bf0 1ed0 01 pmshell
0008  0004 0001 0004 0002 blk 0800 ab59a000 ab9c7020 ab988f60 1ed4 01 pmshell
0009  0004 0001 0004 0003 blk 0800 ab59c000 ab9c7020 ab9891d8 1edc 01 pmshell
000a  0004 0001 0004 0004 blk 0800 ab59e000 ab9c7020 ab9893d0 1eb8 01 pmshell
000b  0004 0001 0004 0005 blk 0800 ab59f000 ab9c7020 ab9895c8 1eb0 01 pmshell
000c  0004 0001 0004 0006 blk 0800 ab5a0000 ab9c7020 ab9898b0 1edc 01 pmshell
000d  0004 0001 0004 0007 blk 0800 ab5a1000 ab9c7020 ab9898da 1edc 01 pmshell
000e  0004 0001 0004 0008 blk 0800 ab5a2000 ab9c7020 ab989fa0 1eb8 01 pmshell
000f  0004 0001 0004 0009 blk 0800 ab594000 ab9c7020 ab989f88 1ea8 01 pmshell
0010  0004 0001 0004 000a blk 0800 ab5a5000 ab9c7020 ab989a18 1eb8 01 pmshell
0011  0004 0001 0004 000b blk 0800 ab5a6000 ab9c7020 ab989a39 1eb8 01 pmshell
0012  0004 0001 0004 000c blk 0800 ab5a7000 ab9c7020 ab989a58 1eb8 01 pmshell
0013  0004 0001 0004 000d blk 0800 ab5a8000 ab9c7020 ab989a78 1eb8 01 pmshell
0014  0004 0001 0004 000e blk 0800 ab5a9000 ab9c7020 ab989a98 1eb8 01 pmshell
0015  0004 0001 0004 000f blk 0800 ab5ba000 ab9c7020 ab989ab7 1ea8 01 pmshell
0016  0004 0001 0004 0100 blk 0800 ab5bb000 ab9c7020 ab989af6 1ed0 01 pmshell
0017  0004 0001 0004 0101 blk 0800 ab5bc000 ab9c7020 ab989a97 1eb0 01 pmshell
0018  0004 0001 0004 0102 blk 0800 ab5bd000 ab9c7020 ab989ab7 1ea8 01 pmshell
0019  0004 0001 0004 0103 blk 0800 ab5be000 ab9c7020 ab989af6 1ed0 01 pmshell
0019  0004 0001 0004 0104 blk 0800 ab5bc000 ab9c7020 ab989a97 1eb0 01 pmshell
0019  0004 0001 0004 0105 blk 0800 ab5bd000 ab9c7020 ab989ab7 1ea8 01 pmshell
0019  0004 0001 0004 0106 blk 0800 ab5be000 ab9c7020 ab989af6 1ed0 01 pmshell
```

qhpsbadapp is non-zero, which indicates that the BadApp dialog has been displayed.

pmqfocus points to a shell thread, in fact the BadApp dialog thread.
The QHPSTRUCT shows Tid 2, Pid e, flags 6 and SGID 16.

.P shows this to be slot 34.

If we use the technique described in How to find the MQ of any thread we will find the MQ for the bad application.

--------------------------------------------

Finding Application and System Queue Elements

This example shows how to find the queue element on both the system queue and an application queue.

A similar technique applies to both types of queue. The system queue header is located from the address at psysqueue. Location of application queue headers has been discussed in How to find the MQ of any thread.

The queue header contains the current read and write pointers, the queue element length and number of elements queued.

We illustrate this with the system queue in the following example:

```c
##dd psysqueue l1
deff:00000000 1bdf0ac0
##dd %1bdf0ac0
%1bdf0ac0 00000000 0030001e 00000078 1bdf0ae4
%1bdf0ad0 1bdf18f4 1bdf1840 1bdf0fd0 00060000
%1bdf0ae0 00070007 00000072 00510196 000002fe
%1bdf0af0 00342420 1c0a9c00 01040040 00335362
%1bdf0b00 00700040 015c0000 000000c1 26cf0000
%1bdf0b10 1c000034 04010ca 53c00104 0000033
%1bdf0b20 00000071 00c1015c 000802fe 003426cf
%1bdf0b30 1c0a9c00 01040040 003353ff 00700040
##dw %1bdf1840
```
MQ+0x4 tells us 0x30 elements are queued, of length 0x1e bytes each.

MQ+0x14 is the current read pointer.

Displaying the queue from the current read pointer we can read off the first few message IDs since they are located at +0x0 of each entry: 70, 71, 72, and so on.

In an application queue the element length is 0x20.

PM Worked Examples Under OS/2 2.x

Dealing with PM application problems under OS/2 2.x is similar to WARP. The principle difference being that the messaging and windowing function in PM is provided by the 16-bit DLL, PMWIN.DLL.

Most of the message structures are analogous to those of PMMERGE.DLL, their layouts are similar.

Under PMWIN.DLL most pointers are either offsets from a predefined segments or selectors. Thus where there are double-word pointers in PMMERGE.DLL structures, there are word length fields in PMWIN.DLL.

The following three symbol files are required for debugging PM applications problems under OS/2 2.x:

- PMWIN.SYM
- PMGRE.SYM
- PMSHAPI.SYM

A selection of useful symbols in the OS/2 2.x PM environment, with their equivalent OS/2 3.0 is listed below:

<table>
<thead>
<tr>
<th>OS/2 3.0</th>
<th>OS/2 2.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmqlist</td>
<td>smqlist</td>
</tr>
<tr>
<td>pmqsystock</td>
<td>smqsyslock</td>
</tr>
<tr>
<td>pmqfocus</td>
<td>smqfocus</td>
</tr>
<tr>
<td>pmqshell</td>
<td>smqshell</td>
</tr>
<tr>
<td>pmqshell2</td>
<td>smqshell2</td>
</tr>
<tr>
<td>pwndfocus</td>
<td>pwndfocus</td>
</tr>
<tr>
<td>fBadAppDialog</td>
<td>fBadAppDialog</td>
</tr>
<tr>
<td>qhpsBadApp</td>
<td>qhpsBadApp</td>
</tr>
</tbody>
</table>

Another significant difference between the two environments is in the calling conventions:

- PMMERGE APIs use the 32-bit C calling convention
- PMWIN APIs use the 16-bit Pascal calling convention

In effect this means that parameters on stacks, and in some control blocks, are stored in reverse order.

There are four symbols that do not have equivalents in PMMERGE, these are:
The selector for the **AAB** regs segment for a process.

The selector for the **SMS** segment.

The table of WND heap pointers.

The PM FastSafe RAMSEM, which is equivalent to the User_Sem PM Semaphore of PMMERGE.

We now run through a brief sequence of examples that illustrate:

- Finding an MQ and AAB registers
- Finding an SMS from an MQ
- Finding the WND from an HWND
- Finding a BadApp process an MQ
- Finding the System Queue

---

**Finding an MQ and AAB Registers**

```plaintext
# .s 8
# .p 8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
*0008* 0006 0001 0006 0001 blk 0500 7b936000 7bb460d0 7bb28a58 1eb8 01 pmshell
# dw winsel l1
fd17:00000032 003f
# dd 3f:0
003f:00000000 00000000 00000000 00000000 00000000
003f:00000010 00081037 0000eb7 f3f0000 00000000
003f:00000020 00000000 00000000 00000000 00000000
003f:00000030 00000000 00000000 00000000 00000000
003f:00000040 00000000 00000000 00000000 00000000
003f:00000050 00000000 00000000 00000000 00000000
003f:00000060 00000000 00000000 00000000 00000000
003f:00000070 00000000 00000000 00000000 00000000
# dd 003f:0
003f:00000000 00000000 00000000 00000000 00000000
003f:00000010 00000000 00000000 00000000 00000000
003f:00000020 00000000 00000000 00000000 00000000
003f:00000030 00000000 00000000 00000000 00000000
003f:00000040 00000000 00000000 00000000 00000000
003f:00000050 00000000 00000000 00000000 00000000
003f:00000060 00000000 00000000 00000000 00000000
003f:00000070 00000000 00000000 00000000 00000000
# dd ebe7:0
ebe7:00000000 001a0000 00640000 0aaa0082 06e80e68
ebe7:00000010 80002ffe 80018001 00010006 06d60001
ebe7:00000020 bf5108b3 02252549 016a0000 00010000
ebe7:00000030 00000000 00000000 00000000 00000000
**ebe7:00000040** 00000000 00000000 00000000 00000000
**ebe7:00000050** 00000000 00000000 00000000 00000000
**ebe7:00000060** 00000010 54530000 0000022a 018c0000
**ebe7:00000070** 1902ec6f 000000ff 00100000 00800000
# dw ebe7:0
ebe7:00000000 00000000 00000000 00000000 00000000
# dd ebe7:0
0000 0000 0000 0000 0000 0000 0000 0000 0000
```

WinSel gives the AAB segment selector.
Note:

WinSel is allocated in instance data, so must be viewed from a thread slot of the process in question.

Each entry is 0x10 bytes, one for each thread of the process.

The first entry is reserved.

The first double word of each entry is the past PM error for that thread and the second double word contains the selector for the MQ of that thread.

The key fields of interest in the MQ are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>chain pointer</td>
</tr>
<tr>
<td>+0x2</td>
<td>Queue element length</td>
</tr>
<tr>
<td>+0x4</td>
<td>number of elements queued</td>
</tr>
<tr>
<td>+0x6</td>
<td>Queue depth</td>
</tr>
<tr>
<td>+0x8</td>
<td>Top of queue</td>
</tr>
<tr>
<td>+0xa</td>
<td>Bottom of queue</td>
</tr>
<tr>
<td>+0xc</td>
<td>Current read pointer</td>
</tr>
<tr>
<td>+0xe</td>
<td>Current write pointer</td>
</tr>
<tr>
<td>+18</td>
<td>Pid</td>
</tr>
<tr>
<td>+1a</td>
<td>Tid</td>
</tr>
<tr>
<td>+1c</td>
<td>SGID</td>
</tr>
<tr>
<td>+30</td>
<td>SMS on which we are waiting a response</td>
</tr>
<tr>
<td>+32</td>
<td>SMS currently dispatched to our window procedure</td>
</tr>
<tr>
<td>+78</td>
<td>SMS at head of received list</td>
</tr>
<tr>
<td>+7e</td>
<td>thread slot id</td>
</tr>
</tbody>
</table>

Finding an SMS From an MQ

```c
##dw smgsyslock 1l
fd9f:000003d4  e55f
##dw e55f:0
 e55f:00000000 e567 001a 0000 000a 0082 0186 0082 0082
e55f:0000010 a400 0006 0006 0006 003d 0009 001e 072a
e55f:0000020 09b3 db25 2549 00f5 0000 005d 0000 0001
e55f:0000030 0094 0000 0010 0000 0000 0000 0000 0000
e55f:0000040 0000 0000 0000 0000 0000 0000 0000 0000
e55f:0000050 0000 0000 0000 0000 0000 0000 0000 0000
e55f:0000060 0000 0000 ec37 5453 061c 0000 0000 6c4c
e55f:0000070 ec6f 0002 0bff 0000 0000 0000 0000 0048
##.a 48
##.p 48
 Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0048# 003d 0006 003d 0009 blk 0500 7b9b6000 7bb51cc4 7bb2f758 1eb8 1e turkey
##dw selsms 1l
fd9f:00001c2a  ec5f
```
The thread with the system queue locked is waiting for a response to **WinSendMsg**. MQ+0x30 has the sent SMS offset. The SMS selector is found from **selsms**.

The key fields in the **SMS** are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>Chain pointer offset</td>
</tr>
<tr>
<td>+0xc</td>
<td>Sending MQ selector</td>
</tr>
<tr>
<td>+0xe</td>
<td>Receiving MQ selector</td>
</tr>
<tr>
<td>+0x1a</td>
<td>Message Parameter 2</td>
</tr>
<tr>
<td>+0x1c</td>
<td>Message Parameter 1</td>
</tr>
<tr>
<td>+0x1e</td>
<td>Message Id</td>
</tr>
<tr>
<td>+0x20</td>
<td>Offset to WND</td>
</tr>
<tr>
<td>+0x22</td>
<td>Selector to WND</td>
</tr>
</tbody>
</table>

In this example the message has been sent to slot 49.

We see that the message has yet to be dispatched since it is still queued on the receive list (MQ+0x78).

Finding a WND From an HWND
In this example we find the WND for the desktop from the HWND which is stored at hwnddesktop.

The HWND comprises an offset concatenated with an identifier, the low order nibble of which is a heap index. Thus, for the desktop:

```
    .    .  .
    .    .  .
    .    .  .
    .    .  .
```

vphheapwnd points to a table of heaps. Each entry is a far pointer and there are at most 16. The index nibble of the HWND is used to select the heap pointer. In this example there is just one entry: ec6f:0000

We use the offset from the HWND with the heap selector to get the PWND. In this case ec6f:13c.

The key fields of interest in the WND are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>Next Sibling WND far pointer</td>
</tr>
<tr>
<td>+0x4</td>
<td>Parent WND far pointer</td>
</tr>
<tr>
<td>+0x8</td>
<td>Child WND far pointer</td>
</tr>
<tr>
<td>+0xc</td>
<td>Owner WND far pointer</td>
</tr>
<tr>
<td>+0x24</td>
<td>MQ selector that services this window</td>
</tr>
<tr>
<td>+0x26</td>
<td>ID and Index portion of the HWND for this WND.</td>
</tr>
<tr>
<td>+0x28</td>
<td>16-bit far pointer to the Window Procedure.</td>
</tr>
<tr>
<td>+0x2c</td>
<td>32-bit pointer to the Window Procedure.</td>
</tr>
</tbody>
</table>

--------------------------------------------

Finding a BadApp Process and MQ

```
##db bbadappdialog l1
%1f8d07ae 01
##dw qhpsbadapp
%1f8d16b8  0002 003d 000a 001e 000c 0000 0000 0000
%1f8d16c8  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16d8  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16e8  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16f8  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1708  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1718  0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1728  0000 0000 0000 0000 0000 0000 0000 0000
```
The fields of the **QHPSTRUCT** are in a different order to the PMMERGE version:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>Flags</td>
</tr>
<tr>
<td>+0x2</td>
<td>Pid</td>
</tr>
<tr>
<td>+0x4</td>
<td>Tid</td>
</tr>
<tr>
<td>+0x6</td>
<td>SGID</td>
</tr>
</tbody>
</table>

---

**Finding the System Queue**

```
##dw mhsysqueue
fd87:00000000 0000 001c 001a 0078 0ad6 17f6 0ee2 11ba
fd87:00000010 0000 0006 0007 0007 0000 0001 0000 ebe7
fd87:00000020 06d6 0008 0ba5 0000 0000 e55f 072a 0048
fd87:00000030 0006 fff6 0000 0000 0000 0000 0000 0588
fd87:00000040 0339 032c 00ac 00bc 00cc 00dc 00ec 00fc
fd87:00000050 010c 012c 011c 013c 014c 015c 0000 0000
fd87:00000060 0072 00f5 005d 02fe 0000 dbc2 2549 8000
fd87:000000ef2 0000 8000 0000 9150 08a8 08a8 0070 0032
fd87:000000f02 00ec 0000 0000 dfaa 2549 0000 c3b7 fff6
fd87:000000f12 0002 0000 0012 08a8 0071 0032 00ec 83fe
fd87:000000f22 0000 e007 2549 fff6 2002 0000 0012 08a8
fd87:000000f32 e007 08a8 0072 0032 00ec 03fe 0000 e065
fd87:000000f42 2549 2006 0000 0016 0000 2006 08a8 08a8
fd87:000000f52 0771 0032 00ec 82fe 0000 e0c3 2549 0000
```

**mhsysqueue** points directly at the system queue.

The current read pointer at +0x0e, is 0x0ee2.

The queue entry length at +0x02, is 0x001c.

Displaying the queue from the read pointer shows the first few elements queued are for message IDs, 72, 70, 71, 72 and so on.

Each entry on the system queue is a **SQMSG**. The key fields are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>Message ID</td>
</tr>
<tr>
<td>+0x2</td>
<td>Message Parameter 1</td>
</tr>
<tr>
<td>+0x6</td>
<td>Message Parameter 2</td>
</tr>
</tbody>
</table>

For an application queue, the entries are **QMSG** structures. The key fields of these are:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0x0</td>
<td>HWND</td>
</tr>
<tr>
<td>+0x4</td>
<td>Message Id</td>
</tr>
<tr>
<td>+0x8</td>
<td>Message Parameter 1</td>
</tr>
<tr>
<td>+0xc</td>
<td>Message Parameter 2</td>
</tr>
</tbody>
</table>
Dump Analysis of Loops in Ring 0 Code

Ring zero loops can sometimes be successfully analysed from a dump. The trick is knowing how to locate the register set at the time the dump was taken.

The Dump Formatter only implements the .R command, which obtains the registers from a stack frame on the thread's ring 0 stack. Under the kernel Debugger there is no problem: the R command will display the current system registers.

Note:
If a thread never runs in User Mode, such as the internal PID 0 threads then a stack frame is never built and .R will be unsuccessful in formatting the registers.

Fortunately there is a way of obtaining the current registers:

When a dump is initiated using Ctrl-Alt-Numlock-Numlock a keyboard interrupt is initiated by the processor hardware.

Via the IDT control passes to the interrupt router who is responsible for switching to the interrupt stack before passing control to the appropriate interrupt handler.

The interrupt router checks to see if the system is already running from the interrupt stack.

If it isn't then an interrupt stack frame is built on the current stack and the stack frame pointer is saved in fpoldstack. Then the SS selector is switched to the interrupt stack selector (E8).

If it is then a nested interrupt has occurred and the interrupt stack frame is built on the interrupt stack itself.

It is from fpoldstack that we are able to obtain the registers before any interrupt occurred. The following debug log illustrates this and many of the techniques previously discussed.

Ring 0 Loop Dump Analysis Example

This example finds a loop in a file system driver from a system dump. For reference, we note the format of the interrupt stack frame as pointed to by fpoldstack as follows:

+0x0       Current interrupt level when prior to interrupt.
+0x4       GS
+0x8       FS
+0xc       ES
+0x10      DS
+0x14      EDI
+0x18      ESI
+0x1c      EBP
+0x20      padesp
+0x24      EBX
+0x28      EDX
+0x2c      ECX
Who's the current thread?

<table>
<thead>
<tr>
<th>Slot</th>
<th>Pid</th>
<th>Ppid</th>
<th>Csid</th>
<th>Ord</th>
<th>Sta</th>
<th>Pri</th>
<th>pTSD</th>
<th>pPTDA</th>
<th>pTCB</th>
<th>Disp</th>
<th>SG</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>00a3</td>
<td>006c</td>
<td>000a</td>
<td>006c</td>
<td>0001</td>
<td>run</td>
<td>0200</td>
<td>7b720000</td>
<td>7bb025c0</td>
<td>7ba8f9e0</td>
<td>0894</td>
<td>25</td>
<td>FRNOLBMG</td>
</tr>
</tbody>
</table>

 Probably a loop of some kind, could be a hot I/O or even dispatcher bug (unlikely).

Where are we?

eax=001fe624 ebx=00002022 ecx=00000029 edx=00000007 esi=00000000 edi=0003e77c


eip=00000179 esp=0003e624 ebp=0003e68c iopl=2 -- -- --

cs=d02f ss=001f ds=0053 es=0053 fs=150b gs=0000


cr2=00000000 cr3=001bb000
d02f:00000179 66ea4102021a5b00 jmp 005b:1a020241

We are in DOSCALL1. Let's see what function was called.

\# dw ss:bp

001f:0000e68c e6bc 0003 ae60 1a02 e77c 0003 e728 0003
001f:0000e69c e70c 0003 0000 0000 0000 0000 0010 0000

\# ln %11134a6d

No Symbols Found

\# m \%11134a6d

hmte=08fe pmte=%fdef0c68 mflags=4498b186 c:\frnv1r0\dll\frnococl.dll

\# u \%11134a6d-10\%11134a5d 085657 or byte ptr [esi+57],dl

%11134a60 6a00 push 00
>> So we're in DOSOPEN

>> We've almost certainly call-gated into the kernel.

>> Check this out ...

# ln %1a02ae1c
%1a02ae1c DOSCALL1 DOS32OPEN

>> Yes, that's where we are.

>> Now let's see if we can find out where in R0 DOSOPEN has got...

# dw interruptlevel l1
0400:000063B2 0000

# dd currentlevel l1
0148:0000529f OS2KRNL DOSOPEN2

>> So, no nested interrupts, but we are handling one
>> (interruptlevel=0000).

>> The current interrupt came from IRQ 1 (currentlevel=1)

>> So a keyboard interrupt, not surprising because the customer was
>> asked to take a dump using ctrl-alt-numlock-numlock, furthermore
>> he obeyed!

>> Lets look at the interrupt stack saved by the interrupt router
>> (prior to switching stacks).

# dd fpoldstack l2
%fff27310 00004b0c 00000030

>> bp is at +1c, sp at +20, cs at +40, eip at +3c
>> We took the dump at when cs:eip=23d0:be93
>> who is this?

# .m 23d0:0
*har par cpg va flg next prev link hash hob hal
0021 %fee132e0 00001400 %fc953000 121 0020 0022 0000 0020 0022 0000 =0000
>> 23d0 is allocated out of the Kernel Swappable Read/Write heap.
>> Lets see who owns this heap block. Need to look at the VMKSHB
>> shared heap block header...

# dg 23d0
23d0  Code    Bas=fca15000 Lim=0000ff5f DPL=0 P RE    A
# dw %face15000-10
%fca14ff0  0000 0000 0000 0000 ff68 5200 ff4d 23d0
%fca15000
Invalid linear address: %fca15000

>> VMKSHB is an 8 byte prefix of the form:
>> ulong size || 0x520000
>> ushort hob
>> ushort sel

>> check the owner hob

# .mo ff4d
ff4d fsd6
#

>> This was allocated by/for the 6th loaded FSD.
>> N.B fsd8 is used for the 8th and subsequent FSDs
>> in the same way ddl6 and fsd-fsd8 system object ids are not used, but
>> instead the hmte for the driver is used. This avoids the problem of
>> the non-uniqueness of ddl6 and fsd8. It also directly identifies
>> the driver module.

>> Who is fsd6?
>> There are two ways to home in on this.
>> First method.

# .lml
hmte=18e2 pmte=%fe1a1000 mflags=0498b188 c:\frnv1r0\dll\frnolgar.dll
hmte=193e pmte=%fe1a13ac mflags=0498b188 c:\frnv1r0\dll\frnosars.dll
hmte=18fd pmte=%fe1a18d4 mflags=4498b186 c:\frnv1r0\dll\frnoutil.dll
hmte=1933 pmte=%fe1a194c mflags=4498b186 c:\frnv1r0\dll\frnofios.dll
hmte=0fc7 pmte=%fdd6f550 mflags=4498b1c6 c:\frnv1r0\dll\frnlmma.dll
.
.
>> 200 hundred lines later
.
.
hmte=0b0a pmte=%f00bb6f8 mflags=0408b1c8 c:\cmlib\dll\acshpres.dll
hmte=0a19 pmte=%f00bb864 mflags=0408b1c8 c:\cmlib\[red].pml
hmte=0af6 pmte=%f00bb8f0 mflags=0408b1c8 c:\cmlib\[red].pml
hmte=0b7f pmte=%f00bbf8c mflags=0408b1c8 c:\cmlib\[red].pml
hmte=09d4 pmte=%f011d44c mflags=0408b1c8 c:\cmlib\[red].pml
hmte=0a8a pmte=%f098e64 mflags=0408b1c8 c:\cmlib\cm20sys.pml
hmte=0a87 pmte=%fe098f5c mflags=0408b1c8 c:\cmlib\cm20sys.pml
hmte=0ac7 pmte=%fe09f5f60 mflags=0408b1c8 c:\os2\dll\times.times
hmte=0ac5 pmte=%fe05f5b4 mflags=0408b1c8 c:\os2\dll\sysmono.fon
hmte=0ac1 pmte=%fe05f5f4 mflags=0408b1c8 c:\os2\dll\sysmono.fon
hmte=0ab9 pmte=%fe05f5f8c mflags=4498b1c5 c:\os2\dll\pmatm.dll
hmte=0324 pmte=%f013f54 mflags=428a1c9 d:\ibm3995\demoifs ifs
hmte=02ff pmte=%fe0fff90 mflags=428a1c9 d:\ibm3995\demoifs ifs
hmte=01bb pmte=%f00dbcb4 mflags=428a1c9 d:\ibm3995\demoifs ifs
hmte=0109 pmte=%fe0beafa0 mflags=428a1c9 c:\os2\dfs.ifs
hmte=0000 pmte=%fe046d3c mflags=428a1c9 c:\os2\dfs.ifs
hmte=0076 pmte=%fe046f68 mflags=428a1c9 a:\mini_fsd.fsd
.

>> FSDs were installed in order, 76, e0, 109, 1b9, 2ff and 324.

>> fsd6 is therefore hmte=324. Lets check for certain:

# .lmo 324
hmte=0324 pmte=%f013f54 mflags=428a1c9 d:\ibm3995\demoifs ifs
seg sect psiz vsiz hob sel flags
0000 0003 d'ba d'ba 0000 2398 8d60 code shr prel rel
>> and yes we find selector 23d0 in object 8 of demoifs.ifs

>> Second method
>> We could approach this from the FSC control block, which is similar
>> in purpose to the DD header chain.
>> The FSC is a table of FSD entry point tables. We might spot the
>> selector in question being referenced in the FSC. If not, we can
>> unwind the R0 stack until we do find a reference.

>> First the FSC. Dump the SAS for the FSC selector

# .a
--- SAS Base Section ---
SAS signature: SAS
offset to tables section: 0016
FLAT selector for kernel data: 0158
offset to configuration section: 001E
offset to device driver section: 0020
offset to Virtual Memory section: 002C
offset to Tasking section: 005C
offset to FSC section: 00C8

--- SAS Protected Modes Tables Section ---
selector for GDT: 0008
selector for LDT: 0018
selector for IDT: 001A
selector for GDTPool: 0100
--- SAS Device Driver Section ---
offset for the first bimodal dd: 0CB9
offset for the first real mode dd: 0000
sel for Drive Parameter Block: 0520
seg for ABiOS prot. mode CDA: 0468
selector for ABiOS real mode CDA: 6800
selector for FSC: 00C8
--- SAS Task Section ---
selector for current PTDA: 0030
FLAT offset for process tree head: FFF29714
FLAT address for TCB address array: FFF2E06A
offset for current TCB number: FFE23A0E
offset for ThreadCount: FFE23A12
--- SAS File System Section ---
handle to MFT PTree: FDE55FB4
selector for System File Table: 00C0
sel. for Volume Parameter Bloc: 0678
sel. for Current Directory Struc: 06A8
selector for buffer segment: 00A8
--- SAS Information Segment Section ---
selector for global info seg: 0428
address of curtask local infoseg: 03B80000
address of DOS task's infoseg: FFF6FFFF
selector for Codepage Data: 06B8
--- SAS RAS Section ---
selector for System Trace Data Area: 0508
segment for System Trace Data Area: 0508
offset for trace event mask: 09D6
--- SAS Configuration Section ---
offset for Device Config. Table: 0D40
--- SAS Virtual Memory Mgt. Section ---
Flat offset of arena records: FFF2C314
Flat offset of object records: FFF2C32C
Flat offset of context records: FFF2C31C
Flat offset of kernel mte records: FFF27E68
Flat offset of linked mte list:
Flat offset of page frame table:
Flat offset of page range table:
Flat offset of swap frame array:
Flat offset of Idle Head:
Flat offset of Free Head:
Flat offset of Heap Array:
Flat offset of all mte records:

FFF273B8
FFF2A768
FFF29CC0
FFF260B0
FFF294D4
FFF294C4
FFF2A770
FFF2BE24

#
>> FSC selector is c8. Now dump the FCS segment.
# dw c8:0
00c8:00000000
00c8:00000010
00c8:00000020
00c8:00000030
00c8:00000040
00c8:00000050
00c8:00000060
00c8:00000070
# d
00c8:00000080
00c8:00000090
00c8:000000a0
00c8:000000b0
00c8:000000c0
00c8:000000d0
00c8:000000e0
00c8:000000f0
# d
00c8:00000100
00c8:00000110
00c8:00000120
00c8:00000130
00c8:00000140
00c8:00000150
00c8:00000160
00c8:00000170
# d
00c8:00000180
00c8:00000190
00c8:000001a0
00c8:000001b0
00c8:000001c0
00c8:000001d0
00c8:000001e0
00c8:000001f0
# d
00c8:00000200
00c8:00000210
00c8:00000220
00c8:00000230
00c8:00000240
00c8:00000250
00c8:00000260
00c8:00000270
# d
00c8:00000280
00c8:00000290
00c8:000002a0
00c8:000002b0
00c8:000002c0
00c8:000002d0
00c8:000002e0
00c8:000002f0
# d
00c8:00000300
00c8:00000310
00c8:00000320
00c8:00000330
00c8:00000340
00c8:00000350
00c8:00000360
00c8:00000370
# d
00c8:00000380
00c8:00000390
00c8:000003a0

03c8
0000
0570
0e3c
09f8
2122
1bec
1f14

0000
0720
0720
0720
0718
0720
0720
0720

0000
01fc
0580
1120
1130
16e4
1dc8
215c

fde1
0720
0720
0720
0720
0720
0720
0720

0b68
0010
0634
0834
1f24
1b10
0c60
22a0

0738
0720
0720
0720
0720
0720
0718
0720

0b6c
05b4
0640
090c
1f6e
1b38
0d70
2294

0738
0718
0720
0720
0720
0720
0718
0720

111c
0fdc
0000
0bb4
0000
0000
060e
1198

0718
0718
0000
0718
0000
0a50
0a50
0a50

25fc
0000
03b0
26bc
0000
01b6
061c
1224

0720
0718
0720
0720
0000
0a50
0a50
0a50

26b0
0000
062c
0000
0000
000e
065c
086e

0720
0000
0718
0000
0a58
0a50
0a50
0a50

117c
0000
137c
0000
0004
04c4
0666
0e0e

0718
0000
0720
0000
0a58
0a50
0a50
0a50

09e6
29b6
2842
297e
2f0c
5029
0000
124e

0a50
0a50
0a50
0a50
0a50
1000
0000
0a50

125a
263e
288a
29c4
34b2
9cab
0602
3500

0a50
0a50
0a50
0a50
0a50
0140
0a50
0a50

299a
278c
28fc
2cf6
34f2
1232
9d8c
0000

0a50
0a50
0a50
0a50
0a50
0a50
0140
0000

29a8
27aa
298c
2cb8
350c
1240
1568
0000

0a50
0a50
0a50
0a50
0a50
0a50
0a50
0000

0000
00be
09ee
0fa4
1998
1fae
2caa
332c

0000
1018
1018
1018
1018
1018
1018
1018

0000
03e6
0c38
1488
1cae
2998
2f90
333a

0000
1018
1018
1018
1018
1018
1018
1018

0090
05da
0db6
1496
1f92
2bf0
2f9e
3950

1028
1018
1018
1018
1018
1018
1018
1018

008a
0766
0dc2
158a
1fa0
2c9c
31c4
3e9c

1028
1018
1018
1018
1018
1018
1018
1018

3fa2
5029
0000
1fbc
0000
01b0
18e0
3ca7

1018
1000
0000
1018
0000
22b0
22b0
22b0

419a
9cab
08aa
4326
0000
1500
51b0
469a

1018
0140
1018
1018
0000
22b0
22b0
22b0

4318
0000
9d8c
0000
0098
7470
25d0
28ac

1018
0000
0140
0000
22a8
22b0
22b0
22b0

4332
0000
2114
0000
0090
1650
344d
279b

1018
0000
1018
0000
22a8
22b0
22b0
22b0

4316
434d
195d
4d20
6d27
5029
84c0
25f0

22b0
22b0
22b0
22b0
22b0
1000
22b0
22b0

28c5
3b30
4310
4ec6
1aab
9cab
17bb
7570

22b0
22b0
22b0
22b0
22b0
0140
22b0
22b0

4343
43c0
6e50
5bc2
43a1
4a70
9d8c
0000

22b0
22b0
22b0
22b0
22b0
22b0
0140
0000

4347
4353
4c10
4359
8740
4a7f
37f0
0000

22b0
22b0
22b0
22b0
22b0
22b0
22b0
0000

0000
00de
02f2
049e
066b
083e
0acf
0c87

0000
23a0
23a0
23a0
23a0
23a0
23a0
23a0

0000
0152
0369
050f
06eb
0982
0b40
0cfb

0000
23a0
23a0
23a0
23a0
23a0
23a0
23a0

0c96
01bd
03d1
0580
075f
09e7
0bab
0d8d

23f8
23a0
23a0
23a0
23a0
23a0
23a0
23a0

8880
0228
042d
05df
07b5
0a4c
0c1f
0e04

23f8
23a0
23a0
23a0
23a0
23a0
23a0
23a0

0e5d 23a0 0ecb 23a0 0f33 23a0 0f92 23a0
5029 1000 9cab 0140 0000 0000 0000 0000
0000 0000 028d 23a0 9d8c 0140 0905 23a0


The FSC starts with an 8 byte header. Word 1 is the length. Each entry is for each FSD starting with fsd2 (fsd1 is OS2BOOT and not used once the kernel is loaded). Each FSD entry comprises a table of far16 pointers. The first two are a) pointer to FSD attributes and b) FSD name. The remaining are the function entry points (See IFS OEM reference). There are 46 of these. In other words the first fsd entry is at c8:8 and ever subsequent entry is every 12 lines of display. fsd 6 entry starts at c3:308.

what's fsd6 called?

The evidently is the optical library fsd.

We didn't find the current cs:eip in the fsd function table so we unwind the r0 stack.

The problem here is that the kernel is not using ebp before calling the fsd. So dump the R0 stack from the last recognisable fsd selector. Look for the first selector that matches one used in fsd6's function table.

The evidently is the optical library fsd.
Finally for future reference the FSD entry structure is as follows:

+0   FS_ATTRIBUTE; /* -> FSD attribute. (in FSD memory) */
+4   FS_NAME;      /* -> FSD name.      (in FSD memory) */
+8   FS_ATTACH;    /* DosQFsAttach, DosFsAttach */
+1c  FS_CHDIR;    /* DosChdir */
+20  FS_EXIT;     /* DosExit */
+24  FS_FILEATTRIBUTE; /* DosFileInfo, DosSetFileMode */
+28  FS_FILEINFO;  /* DosQFileInfo, DosSetFileInfo */
+2c  FS_FILEIO;   /* DosFileIO */
+30  FS_FINDCLOSE;/* DosFindClose */
+34  FS_FINDFIRST;/* DosFindFirst */
+38  FS_FINDFROMNAME; /* DosFindFromName_Private to server */
+4c  FS_FINDNEXT; /* DosFindNext */
+40  FS_FINDNOTIFYCLOSE; /* DosFindNotifyClose */
+44  FS_FINDNOTIFYFIRST; /* DosFindNotifyFirst */
+48  FS_FINDNOTIFYNEXT; /* DosFindNotifyNext */
+4c  FS_FSINFO;   /* DosQFsInfo, DosSetFsInfo */
+50  FS_INIT;     /* -- No corresponding API */
+54  FS_IOCTL;    /* DosDevIoctl */
+58  FS_MKDIR;    /* -- No corresponding API */
+60  FS_MOVE;     /* DosMove */
+64  FS_NEWNAME;  /* DosNewname */
+68  FS_NMPPIPE;  /* All named pipe related API's */
+6c  FS_OPENCREATE; /* init time only */
+70  FS_PATHINFO; /* DosQPathInfo, DosSetPathInfo */
+74  FS_PROCESSNAME; /* -- No corresponding API */
+78  FS_READ;    /* DosRead, DosReadAsync */
+7c  FS_RMDIR;    /* DosRmdir */
+80  FS_SETSWAP;  /* -- No corresponding API */
+84  FS_WRITE;   /* DosWrite, DosWriteAsync */
+88  FS_OPENPAGEFILE; /* init time only */
+8c  FS_ALLOCATEPAGESPACE; /* size swap file */
+90  FS_CANCELLOCKREQUEST; /* DosCancellockRequest */
+94  FS_FILELOCKS; /* DosSetFileLocks */
+98  FS_VERIFYUNCNAME; /* Used to save function addresses */
+9c  FS_COMMIT;  /* DosBufReset, DosClose */
+a0  FS_DOPAGEIO; /* perform paging */
+a4  FS_FSCtl;    /* DosFsCtl */
+a8  FS_FLUSHBUF; /* DosBufReset */
+ac  FS_SHUTDOWN; /* DosShutdown */
+b0  FS_SDCALLFILEPTR; /* Used to save function addresses */
+b4  FS_SDFileSystem; /* at shutdown time. These functions */
+b8  FS_SDRead;   /* will only be called by shutdown */
+bc  FS_SDWrite;  /* filters. */

* Bit masks for FS_ATTRIBUTE (remember FS_ATTRIBUTE points to the attribute) * word rather than containing it directly. *)

* FS_ATTR_REMOTE 0x0001 /* 0 = local FSD, 1 = remote FSD */
* FS_ATTR_UNC 0x0002 /* 0 = normal, 1 = this is UNC FSD */
* FS_ATTR(lockINFO 0x0004 /* 0 = no notice, 1=notify filelocks */
* FS_ATTR_LVL7 0x0008 /* 0 = no level 7 requests, 1 = yes */
* FS_ATTR_PIPEsvr 0x0010 /* 0 = don't FSD on PIPE req, 1 = yes */
* FS_ATTR_VERNO 0x0000 /* bits 28-30 version no */
* FS_ATTR_VERNO 0x0000 /* bits 28-30 version no */
* FS_ATTR_EA 0x8000 /* bit 31 -> 1 = extended attribute */
* FS_ATTR_COMMIT_ALL 2 /* all handles commit */
* FS_ATTR_COMMIT_ONE 1 /* one handle commit */
* FS_CL_ORDINARY 0 /* ordinary close */
* FS_CL_FORPROC 1 /* final close for process */
* FS_CL_FORSYS 2 /* final close for system */
The Kernel Debugger is essentially a replacement OS/2 Kernel module that contains an in-built debugger component. With the debugger one may halt system execution, inspect and alter memory and registers and display system control blocks. The debugger is controlled from a dump ASCII terminal (the debugging console) which is connected to the machine under test (MUT), either directly or via a modem-modem link, through one of its COMx ports. The debugger supports a comprehensive command set, which is fully described in the *Kernel Debugger and Dump Formatter Command Reference*.

The debug kernel is distributed in two forms:

**ALLSTRICT**

This version of the kernel contains all optional self diagnostic (otherwise known as strict or assertion checking) code. Besides this functional difference many of the system control blocks have extra accounting and signature fields. This has a number of consequences that may affect problem diagnosis:

1. Performance characteristics will be different since extra checking and accounting is being performed.
2. Memory usage will be different because of extra diagnostic code, extensions to system control blocks and in some cases additional space to cause page faults rather than overlays by erroneous code.
3. Timing critical problems might not be re-creatable under the ALLSTRICT kernel.
4. Secondary problems may be detected or even introduced through the use of additional diagnostic code.

**HSTRICT**

This version of the kernel is essentially the RETAIL kernel with the debugger component. It contains only a limited set of strict checking code. The system control blocks are of the same form as those used by in the RETAIL kernel. The performance characteristics of the HSTRICT kernel are closer to those of the RETAIL kernel than the ALLSTRICT kernel. For this reason the HSTRICT kernel is recommended as a first choice when diagnosing application and non-system problems.

The base version of the ALLSTRICT kernel is distributed with the OS/2 Developer's Toolkit. Versions of the HSTRICT and ALLSTRICT kernels for fix packs may be obtained from the following sources:

- The OS/2 Base Product CDROM for WARP is distributed with the ALLSTRICT kernel and Dump Formatter. (For the initial release of WARP this was only available on the US version of WARP).
- The Developer Connection CDROM - this may be ordered through the Developer Assistance Program (DAP) or the System Library Subscription Service (SLSS).
- From your local IBM Marketing Representative.
- From the World Wide Web at URL:

  ftp://service.boulder.ibm.com

- For IBM customers from by FTP from the node:

  ftp.software.ibm.com, directory ps\products\os2\fixes\debug

- For IBM internal users by FTP to the SDM at node:

  sdm.austin.ibm.com

Logon using Id and Password Anonymous. A list of files is contained in files.bbs.

**Note:**

The Kernel Debugger packages obtainable from the SDM are equipped with an installation procedure and two text files dbsetup.txt and modemset.txt that give instructions on how to install the debugger for local and remote debugging via a modem.
Kernel Debugger Local Set-up

The following items are required to install and set up a local debug session:

- Either the HSTRICT or ALLSTRICT kernel appropriate to the level of the MUT.
- System symbol files. These are optional, but useful breakpoints and system data are difficult to locate without them.
- Application symbol files. These are only necessary if you intend to debug complex applications where data and subroutines are difficult to locate without them.
- System Trace definition and Formatting files. These are only required if you intend to trace kernel dynamic tracepoints while using the debug kernel.
- A null modem cable.
- An asynchronous ASCII dumb terminal or an emulator on another PC. Softerm, which is distributed with OS/2 is suitable. PMDF, which is part of the OS2PDP package distributed with this book also provides a terminal emulator interface suitable for use with the Kernel Debugger. Other popular emulators used with the Kernel Debugger include: PMDEBUG, DEBUGO and LOGICOMM.

Confusion sometimes arises over the installation of the kernel debugger, particularly as the OS/2 Developer's Toolkit distributes debug versions of other OS/2 modules. Note in particular:

- The debug versions of OS2LDR, PMDD.SYS, PMGRE.DLL and PMWIN.DLL are optional. These modules will route additional diagnostic information to the debug console if they are installed.
- No modification of CONFIG.SYS is required.
- A secondary console attached to the MUT may not be used as a debug console.

Installing the Debug Kernel

If you use the OS/2 Developer's toolkit to install the debug kernel then the installation is performed automatically using the supplied DBGINST command. If you choose to install the debug kernel manually then perform the following steps:

1. Copy the debug kernel (OS2KRNLD or OS2KRNLB) to the root directory of the boot drive.
2. Copy the symbol files into the same directories as their corresponding load modules. Usually system symbol files are distributed on a diskettes that have the same directory structure as OS/2 system code. This conveniently allows the UNPACK command to be used to copy all symbols files in one operation (per diskette).
3. Unhide the RETAIL kernel module using the following command:
   ATTRIB -r -s -h OS2KRNL
4. Rename the RETAIL kernel to something unique, e.g OS2KRNLR
5. Rename the ALLSTRICT or HSTRICT kernel to OS2KRNL. There is no need to hide or make the replaced kernel read-only, unless you wish to protect yourself against accidents!

The MUT is now ready to use in debug mode as soon as it is re-booted. Before that happens the debug console needs to be set up.

Note:

It is possible to run the MUT with the debug kernel installed without setting up the debug console. This particularly useful when diagnosing pervasive problems. If the COM port settings are correct when the problem reoccurs then the debug console may be connected at that time.
Debug Terminal Set-up

This section describes the connection and set-up of the debugging console. You may need to know the operational requirements of both your local COM port (on the MUT) and dumb ASCII terminal. Fortunately the debug kernel does not impose any form of hand-shaking or a fixed COM speed setting. In many cases default settings apply. First we discuss the cable requirements.

A null modem cable is required to connect the MUT to the debug console. This is essentially a 3-wire circuit that connects the two COM connectors together. Some PCs are equipped with a 25-pin sockets, other 9-pin. A null modem cable is a symmetric circuit so we do not distinguish which is the MUT and which the console.

25-to-25 Pin Cable

<table>
<thead>
<tr>
<th>MUT/CONSOLE</th>
<th>CONSOLE/MUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB25J</td>
<td>DB25J</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

25-to-9 Pin Cable

<table>
<thead>
<tr>
<th>MUT/CONSOLE</th>
<th>CONSOLE/MUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB25J</td>
<td>DB9J</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

9-to-9 Pin Cable

<table>
<thead>
<tr>
<th>MUT/CONSOLE</th>
<th>CONSOLE/MUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB9J</td>
<td>DB9J</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Note:

The three connections involved are:

- RX (receive)
- TX (transmit)
- SG (Signal Ground)
The null modem cable essentially connects RX-TX and SG-SG. The pin conventions for RX and TX on a 25-pin connector reverse those of a 9-pin connect. Thus the 25-9 connection looks like a non-null circuit.

If you intend to debug on a number of different set-ups then it is worth equipping yourself with the following items, which are commercially available:

- Standard modem cable.
- A gender changer.
- A null modem converter.
- A 25-9 pin converter.

With these items you should be able to cater for most variations and remote connection as well.

The next thing to consider is the COM port settings. By default the debug kernel will first select COM2. It that is in use then COM1. If you require the debugger to use another COM port, or a non-standard I/O port address then you might need to set this explicitly by using the `.B command`, which should be entered in the KDB.INI initialisation file.

By default the kernel debugger initialise the selected COM port to run at 9600 bits per second. If your debugging console requires a different speed setting then you should convey this to the debug kernel using the `.B command`, again entered in the KDB.INI file.

The default communications protocol uses 8 data bits, 1 stop bit and no parity. If this needs to be different then it may be set using the `.O command` also entered in the KDB.INI file.

Finally some COM ports require the DTR signal to be held high before allowing communication. If this is necessary then it can be set using the debug kernel to write to the I/O port that controls the COM port set-up register. This may be done using the the `.O command` entered in the KDB.INI file.

Examples of using these commands in KDB.INI is given in the next section.

Having set up the COM port requirements on the MUT the debug console must be set up to match. Precisely how this is done will depend on whether a dumb terminal or terminal emulator software is used. If you use emulator software under OS/2 you may need to use the OS/2 MODE command to select compatible COM port settings for the debugging console's COM port.

The KDB.INI Initialisation File

The debug kernel normally only accepts commands entered at the debugging console. However, during system initialisation it will accept commands entered into a text file, which if used, must be called KDB.INI and reside in the root directory of the boot drive.

The KDB.INI file is read after the kernel has loaded and the kernel symbols are loaded and the system is running in protect mode.

Warning:

The content of the KDB.INI file is somewhat sensitive. If you make a syntax or format error then you may hang the system and have to re-boot from installation diskettes to recover.

On most systems the use of a KDB.INI file is not required to establish correct operation of the COM port and should be avoided.

Each command must be terminated with a <CR> <LF> pair except the last in the file.

The KDB.INI is most easily created using:

COPY CON: KDB.INI

Enter the commands you require, using the <RETURN> key after each command except the last. For the final command, terminate it using the sequence: Ctrl-Z <RETURN>.

Note:

Use of an editor for creating KDB.INI may not be suitable if the <CR> <LF> sequence cannot be suppressed from the last line.
The following example shows how to select COM3 at 1200 bps, with DTR held high and to prepare the debugger to intercept any ring 2 or 3 traps.

```
.b 1200t 3e8
O 3ec 1
vsf *
g
```

Notes:

Since the default arithmetic base for the debugger is hexadecimal a suffix is required if the COM port speed is specified in decimal as in the example.

We have assumed a standard port address assignment for COM3, namely 3e8 for data register and 3ec for control register.

The VSF command causes the debugger to intercept all ring2 and ring3 traps and give control to the debug console.

The G command is required unless you want to enter the debugger as soon as the kernel has entered protect mode, loaded its symbol file and executed the KDB.INI file.

--------------------------------------------

Kernel Debugger Remote Set-up

This section describes how to use the kernel debugger remotely, that is with a modem-modem link between the machine under test (MUT) and the debugging console.

The first step is to install the debug kernel and symbols files on the MUT as described preceding section, Kernel Debugger Local Set-up.

Although the Debug Kernel will work with nearly any modem, configuration details are unique to each modem. This topic describes the setup of several modems, and gives general guidelines for setting up others.

--------------------------------------------

Items Required to Setup a System for Remote Debugging

To complete the installation, you will need:

- The RETAIL and either the HRICT or ALLRICT Kernel
- A modem
- A modem data cable
- An analog dial-in telephone line
- Communications software.

--------------------------------------------

Modem

Most asynchronous modems currently available will be suitable for use as a remote-debug modem. For best performance, the modem should:

- Support auto-answer operation
- Support locked DTE speed at 9600 bps
• Allow connections at CCITT V.32 (9600 bps), and V.22bis (2400 bps)
• Support error-correction (MNP or V.42)
• Save configuration so a power-outage does not lose settings.

Modem data cable

The configuration of the cable used to connect the modem to the MUT is not important. Any serial data cable should have the connections required by the debug kernel. Just make sure you don’t use a null-modem cable. You will either need a 25-to-25 pin cable (for connection to the built-in serial port on a PS/2), or a 25-to-9 pin cable (for connection to a 9-pin serial port).

Required connections for remote debug cable:

25-to-25 Pin Cable

<table>
<thead>
<tr>
<th>MODEM</th>
<th>COMPUTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB25P</td>
<td>DB25J</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

25-to-9 Pin Cable

<table>
<thead>
<tr>
<th>MODEM</th>
<th>COMPUTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB25P</td>
<td>DB9J</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

Notice the 25-to-9 pin cable reverses pins 2 and 3. Do not confuse this with a null-modem cable - the signals on a 25-to-9 pin cable are normally reversed.

Analog Dial-in Telephone Line

In order to call the modem and connect to the MUT, you will need a standard voice-grade telephone line that can be direct-dialled. A connection can be made if the line must go through a switchboard, but it makes it more difficult for the person doing the debugging. Digital telephone lines won’t work at all with the modem.

Communications Software

Any terminal software that can communicate at 9600 bps will do. OS/2 2.0 comes with a program (Softerm Custom) that is adequate. PMDF, which is part of the OS2PDP package on the CDROM that accompanies this book, also provides a terminal emulation facility but in addition
provides REXX support that allows Kernel Debugger command sequences to be automated.

The Configuration Process

After you have assembled the required items, follow these steps to prepare the MUT for remote debugging:

1. Connect the Modem to the MUT.

   Connect one end of the data cable to the modem, and the other end to the serial port on the MUT. If the MUT has more than one serial port, connect the cable to the port configured as COM2 (the debug kernel uses COM2 by default). On PS/2 systems, the reference diskette can tell you which port is configured as COM2. Connect the telephone line to the modem, and power the modem on.

2. Program the modem for DEBUG operation:

   Programming the modem may be a complex process, depending on the type of modem and the intended use. There are two ways to program the modem:
   - Quick programming for single debug use
   - Full programming for "permanent" debug use.

   The "quick" method is simple, but the modem will not be programmed to recover from loss of power or repeated calls. The "full" method allows the modem to be programmed once, and then used whenever debugging is needed.

   The "quick" programming is performed by the debug kernel itself through use of the KDB.INI file. In addition to containing start up commands for the debugger KDB.INI can also contain modem initialisation strings coded as operands to the Kernel Debugger ? command. For this reason, the modem must be connected and powered on when the MUT is booted, and cannot be powered off until debugging is complete.

   The first lines of the KDB.INI may will be COM port selection and parameters if defaults are not suitable, for example:

   [.B 1200t 1]
   (Set debugger for 1200 bps, comm port 1)

   Following this are the modem initialisation strings, which are unique to each type of modem. The commands in the initialisation string must:
   - Activate "auto-answer"
   - Lock the DTE at 9600 bps
   - Activate XON/XOFF flow control
   - Ignore the DTR signal (not supplied by the debug kernel)
   - Suppress result codes.

   The remaining lines of the KDB.INI file may contain other debugging commands. The last of these is normally G.

   The "quick" programming strings for several popular modems are as follows.

   ? "AT&F E0 Q1 &B1 &H2 &I2 &D0 S0=1"
   US Robotics HST and Dual Standard

   ? "AT&F2 E0 Q2 &D0 &K4 S0=1"
   Supra FAX/Modem V.32bis

   ? "AT&F E0 Q1 &D0 \Q1 S0=1"
   Intel 14.4EX

   An alternative "quick" technique for entering the Hayes modem initialisation commands, which avoids the use of KDB.INI is illustrated by the following example. This example assumes that the default COM2 port is to be used:

   1. In CONFIG.SYS add the following line

      RUN=C:\OS2\CMD.EXE /K C:\MODEM.CMD
2. Edit a file called MODEM.CMD and enter the following two lines

```
MODE COM2:9600,N,8,1
COPY MODEM COM2
```

3. Edit a file called MODEM and enter the following line

```
AT&K4&D0S0=1&W
```

To use "Full" programming, you will configure the modem with the same features as in "quick" programming, but the settings will be stored in the modem's firmware (or set in modem switches). Determining how to store these settings can be difficult. A thorough study of the modem manual may be required. To program the modem, use a terminal emulation program (for example, the SOFTERM program that is supplied with OS/2). When programming the modem, set the terminal program for 9600 BPS operation, and type the appropriate modem string. Since the initialisation string instructs the modem to suppress result codes, the modem will not return a response. The "FULL" programming strings for several modems are:

```
AT&F &B1 &H2 &I2 &W
```

US Robotics HST and Dual Standard

```
AT&F2 E0 Q2 &D0 &K4 S0=1 &W
```

Supra FAX/Modem V.32bis

```
AT&F E0 Q1 &D0 \Q1 S0=1 &W
```

Intel 14.4EX

NOTE: The US Robotics HST Dual Standard does not store all settings, but has external switches instead. After programming the modem, set the switches as follows:

1=ON
   (DTR forced ON)

2=don't care
   (result code type)

3=OFF
   (result code suppressed)

4=ON
   (command echo suppressed)

5=OFF
   (auto-answer enabled)

6=don't care
   (carrier detect function)

7=ON
   (result code in originate mode only)

8=ON
   (AT commands enabled)

9=ON
   (don't disconnect for +++)

10=OFF
   (load NVRAM at power-on)

QUAD=OFF
   (normal connect - ON if null modem cable used)

Once the modem is connected, and programmed, the system should be ready for remote debugging. Re-boot the system with the debug kernel installed. When the telephone rings, the debug modem should answer the phone, and establish connection with the caller. The modem-to-kernel speed should remain at 9600 bps (the default speed used by the debug kernel), but the modem-to-modem speed can be whatever is used by the remote modem. If both modems support error correction, correction will be used.

--------------------------------------------

Using Low-speed Modems
If a 9600 bps modem is not available, a slower modem can be used with the debug kernel. If the modem supports "speed conversion" (a 2400 bps modem with error-correction and compression will support speed conversion), setup is straightforward. Construct the proper initialisation string for the modem, making sure that the modem's DTE speed (modem-to-debugger) speed is locked at 9600 bps. If the modem does not support "speed conversion," construct an initialisation string for the modem, and create a KDB.INI file that resets the debugger to the speed supported by the modem. For example, .B 2400t 2 for a 2400 bps modem. In this case, the person calling the debugger will have to use the speed supported by the modem.

Limitations of This Setup

Since the modem communicates with the MUT at 9600 bps, but can communicate with the remote modem at any speed, the modem must use flow control to avoid data overruns. The only flow control supported by the debug kernel is XON/XOFF. The only problem this causes is when the remote user wants to pause a continuous data display by pressing CTRL-S. If the modem has also sent a CTRL-S, the one from the user will be ignored. You may have to press CTRL-S several times before the display pauses. This is not a problem if the remote user's communications program supports a "scroll-back buffer," in which case there is no reason to pause the display with CTRL-S.

Troubleshooting

If, after following these directions, you cannot establish a remote debug connection, this guide may help:

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modem rings, but doesn't answer</td>
<td>Modem not set for auto-answer</td>
<td>check modem programming (look for AA light on modem).</td>
</tr>
<tr>
<td>... &quot; ... &quot; ...</td>
<td>Phone line not connected to modem</td>
<td>Plug in telephone line to modem.</td>
</tr>
<tr>
<td>Modem answers, but no response from debug kernel</td>
<td>Retail kernel installed</td>
<td>Remove RETAIL kernel and install DEBUG kernel</td>
</tr>
<tr>
<td>... &quot; ... &quot; ...</td>
<td>Data cable not connected properly</td>
<td>Connect data cable from modem to MUT. Plug into COM2 if MUT has more than one serial port</td>
</tr>
<tr>
<td>User at the remote modem sees &quot;garbage&quot; on screen, unable to control debug session</td>
<td>Modem not locked at 9600 bps</td>
<td>check modem configuration</td>
</tr>
<tr>
<td>... &quot; ... &quot; ...</td>
<td>Debug Kernel not operating at 9600 BPS</td>
<td>Add .B 9600T to KDB.INI file (create file if needed, in root directory of boot drive). Re-boot MUT.</td>
</tr>
</tbody>
</table>

Controlling the System From the Debugging Console
Having set up the Kernel Debugger for a Local or Remote debug session the system is ready to be controlled from the debugging console. The console is used in two modes, which for convenience we refer to as:

Monitor mode, and

Command mode

In Monitor mode the console acts merely as a output device for displaying diagnostic messages from the debug kernel and debug versions any other of system modules that write messages to the debugger's COM port. In this mode it is not possible to enter Kernel Debugger commands without having first switched to command mode. In monitor mode the system runs more or less as a retail system except for the performance overheads imparted by the additional diagnostic code.

Monitor mode is in effect initially unless a KDB.INI file is defined.

The console switches to monitor mode after the G command is executed.

In Command mode normal system execution is suspended. The debug component of the kernel monitors the debugging console for command input and indicates this with using one of the following command prompts:

> Signifies that the system has been suspended while in real mode.

# Signifies that the system has been suspended while in protect mode with paging disabled.

- Signifies that the system has been suspended while in V86 mode with paging disabled.

## Signifies that the system has been suspended while in protect mode with paging enabled.

-- Signifies that the system has been suspended while in V86 mode with paging enabled.

In addition to these prompts the Kernel Debugger also uses a data prompt when a commands require additional input. This is signified by a single colon prompt :. Commands such as R and E may use a data prompt.

Command mode is entered when one of the following events occur:

A fatal exception while executing in ring 0

Any unrecoverable exception occurring in a device driver, file system driver or the OS/2 kernel will result in a fatal error if it is allowed to be intercepted by the system exception handlers. When this occurs it is usually not possible to restore the system to a running state.

The VTF command may be used to intercept potentially fatal exceptions before the system's exception handlers receive control. If the exception condition is corrected manually then the system may continue to run after the G command is entered. See Trap and Exception Processing for further information.

An Internal Processing Error (IPE) occurs.

Internal processing errors are unrecoverable conditions that are detected by the OS/2 kernel. Some of these are exceptions (described in the previous bullet); others are inconsistencies that arise from invalid logical conditions or invalid system data. Under the retail kernel IPEs result in the system halting. Under the debug kernel the console enters command mode after an error message is displayed. IPE messages may be suppressed from displaying as a hard error popups by setting the byte at symbol: fDebugOnly to a non-zero value. Under the debug kernel some IPEs are generated for recoverable conditions and allow the system to continue execution after the G command is entered. An example of a recoverable IPE is where the loader detects a bad or mismatched symbol file for a module it is loading. When this occurs the system displays message:

Internal Symbol Error

Command mode is entered. If the G command is subsequently issued the system will be allowed to continue execution without the bad symbol file being activated.

A sticky breakpoint fires.

Sticky breakpoints are set using the BP and BR commands. The system is may be returned to a running state after the G command is entered.

An unhandled non-maskable interrupt (NMI).

NMIs normally signal hardware error conditions. Under the RETAIL kernel these usually result in TRAP 2 fatal exceptions unless an NMI handler has been registered by a device driver. Under the debug kernel, unhandled NMIs cause control to be given to the debugging console from which it is possible to return the system to a running state using the G command.

NMIs are may be generated from several sources, which include:
Channel Check
This occurs when an I/O card activates the channel check signal.

Memory parity error
This occurs when memory capable of parity bit generation, detects a parity discrepancy as memory is fetched from RAM.

DMA bus time-out.
This occurs when a DMA-driven device uses the bus for longer than the maximum allowed period of 7.8 microseconds.

The watchdog timer interrupt.
This occurs when the NMI watchdog (NWD) is enabled and timer interrupts (IRQ 0) are disabled causing loss of timer ticks. OS/2 maintains an NWD count, which if exceeds a maximum value then an IPE is generated. Some hardware/BIOS also maintains an NWD counter, but the precise details of the NWD mechanism are machine specific. For some systems the NWD may not be supported. For further information refer to the appropriate hardware and BIOS reference literature for the machine type under consideration.

Unless the NMI is masked off using by setting mask bit 0x80 in I/O port 0x70, the NMI channel check provides a means of breaking into the system even when it is disabled for (maskable) interrupts, that is, when the CLI instruction has been used to clear the interrupt flag in the EFLAGS register. An ISA-bus system a prototype card may be used to implement the following circuit, which provides an NMI push-button switch:

---

1. A1
   - (NMI Push switch)

2. B1
   - (Ground)

Note:
OS/2 normally only disables NMIs during system initialisation and when the Kernel Debugger is running in command mode. However, the Kernel Debugger will allow only one attempt to break in using a channel check NMI, after which NMIs are disabled until the system is re-booted.

An INT 3 instruction is executed
INT 3 instructions are used by the system to implement tracing (see The System Trace Facility) and software breakpoints. However any program may use INT 3 instructions freely under the Kernel Debugger to cause system execution to be suspended and the debugging console to switch to command mode.

Note:
Under the RETAIL kernel, INT 3 instructions other than those implemented by the system for tracing cause code to be terminated with a TRAP 3 exception.

The user enters Ctrl-C from the debugging console.
Unless the system is in a disabled state, the user may type Ctrl-C from the debugging console at any time to cause immediate suspension of normal system execution and the console to switch to command mode.

The user holds down the r-key from the debugging console at system initialisation time.
If the r-key is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered real-mode for the first time. At this time no symbols have been loaded, paging has never been enabled and the KDB.INI file has not been processed.

Note:
In real-mode many of the Kernel Debugger external commands are not available (because the rely on Virtual Memory Management to be initialised). Attempts to use them may cause unpredictable results or even total system failure.

The user holds down the p-key at the debugging console at system initialisation time.
If the p-key is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered protect-mode for the first time. At this time no symbols have been loaded, paging is disabled and the KDB.INI file has not been processed.

The user holds down the Space-bar from the debugging console at system initialisation time.
If the space-bar is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered protect-mode and fully initialised. At this time OS2KRNL symbols have been loaded and paging is enabled but the KDB.INI file has not been processed.

The KDB.INI file is processed.

If the KDB.INI file is present then the Kernel Debugger effectively enters command mode by executing Kernel Debugger commands from the KDB.INI file. After the last command is executed, the command prompt appears at the debugging console, unless that last command was a G command.

Controlling Output to the Debugging Console

In both monitor and command mode the following control key sequences are supported:

Ctrl-C
Will cancel the currently running command and return the console to command mode.

Ctrl-S
Will temporarily suspend output to the debugging console and suspend system execution.

Ctrl-Q
Will resume system execution and output to the debugging console.

Note: Ctrl-Q and Ctrl-S correspond to the ASCII asynchronous communications control characters: XON and XOFF. These may be used by any terminal emulator, which interfaces with the the Kernel Debugger, as a data pacing mechanism.

Optional System Diagnostic Facilities

Several system components implement optional diagnostic facilities under the debug kernel. These cause additional checking and in some cases detailed information to be displayed at the debugging console when certain debug flags switches are set.

Note:

Debugging switches are not a formally architected feature of the OS/2 operating system. They are provided primarily for use by OS/2 developers in debugging and testing the system. They are therefore subject to change or withdrawal without any notice whatsoever.

In this section the following logging facilities are described:

Forcing a System Dump from the Kernel Debugger
Virtual Memory Management Lock Trace
Virtual Memory Heap Validation
Loader Logging Facility
DosDebug Logging Facility
DosPTrace Logging Facility

Forcing a System Dump from the Kernel Debugger

Sometimes the situation arises where neither a kernel debug session or a system dump alone are sufficient to analyse a problem. Typically
this occurs with problems where evidence of the cause has been removed from the system before the problem occurrence becomes
recognised but the problem itself requires lengthy analysis even when the causal conditions are intercepted. Examples of this are problems where:

Storage overlays, may not be noticed until the valid owner of the storage traps at some later time.

A program terminates apparently normally, but unexpectedly.

A deadlock or hang occurs because a resource owner forgets to release ownership of a shared resource.

If the problem is such that there are readily identifiable criteria that allow it to be intercepted closer to its cause, for example by using
breakpoints under the Kernel Debugger, then being able to take a dump at such a point can be advantageous.

The simplest technique for initiating a system dump is to enter the .SYSDUMP command, which is new from fix pack 29 for Warp 3.0 and
base Warp 4.0. Prior to these releases other techniques have to be employed. The simplest of these is to type the dump key sequence
(Ctrl-Alt-Numlock-Numlock or Ctrl_Alt_F10_F10) from the keyboard of the system under test while the debugger is in console mode. Then
type the G command from the debug console. The keyboard interrupt will be serviced and the standalone dump procedure initiated.

In an unattended situation a manually initiated dump may not be feasible. The following techniques discuss how to initiated the system dump
in a more automated fashion. In some cases it may be possible to set up the command automation from the KDB.INI initialisation file.

The system dump is initiated when the kernel routine RASRST (RAS restart) is called. Normally this occurs from ring 0 when exception
management intercepts a trap and TRAPDUMP is coded in the CONFIG.SYS file or when the keyboard device driver (KDB.SYS) intercepts
a Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 sequence. From ring 3 RASRST is called indirectly via the Dos32ForceSystemDump
API since RASRST is not addressable from any user code selectors. The Kernel Debugger G command allows an address to be specified
where execution is to continue from, which provides a means calling the system dump routine from the debugging console. Before using this
technique, the following points must be understood:

RASRST is not addressable from user code selectors since they have an upper address boundary of at most 512M.

RASRST requires to be executed using a 16-bit code selector.

RASRST requires a ring 0 stack selector to be active.

Dos32ForceSystemDump requires a 32-bit code selector, such as 5b.

On some early versions of OS/2 2.1 Dos32ForceSystemDump is unreliable.

The symbol Dos32ForceSystemDump occurs in both DOSCALL1.DLL and the callgate entry point in OS2KRNL.

From ring 0 the following command will generally be successful in initiating a system dump:

```
g =rasrst
```

From ring 2 or ring 3, 32-bit code the following commands will be successful providing Dos32ForceSystemDump is working correctly. The
address of DOSCALL1:DOS32FORCESYSTEMDUMP is determined first, then a call to Dos32ForceSystemDump is made:

```
ln dos32forcesystemdump
%1a027c78 doscall1:FLAT32:DOS32FORCESYSTEMDUMP

g =1a027c78
```

For 16-bit application code the CS register must be to to a value that will address DOSCALL1:DOS32FORCESYSTEMDUMP. A suitable
selector would be 5b for ring-3 code and 5a for ring-2. So, for 16-bit code this procedure becomes:

```
ln dos32forcesystemdump
%1a027c78 doscall1:FLAT32:DOS32FORCESYSTEMDUMP
r cs 5b  (or r cs 5a)
g =1a027c78
```

If TRAPDUMP is in effect then a dump can be forced by causing an immediate trap. The most effective way to achieve this is to set the
current SS selector to 0 using the R command. For example:

```
r ss=00
g
```
If you wish to trap an application the very next time it runs in user mode then use `.R` to determine the user registers and set a breakpoint on `CS:EIP` in the context of the application's thread slot and specify that `SS` be set to zero when the breakpoint fires. For example:

```
p 2d
Slot  Pid  Ppid Csid Ord Sta Pri  pTSD  pPTDA  pTCB  Disp SG Name
002d  000b  0002  000b  0001 blk 0200  7b700000  7b8c68fc  7b8ac6b0 1eb8  14 mrfilepm
```

```
##.r 2d
eax=00000000 ebx=00000000 ecx=0000aa37 edx=0000a9ef esi=00090bff edi=00090000
eip=00002727 esp=0000b228 ebp=0009b230 iopl=2 -- -- nv up ei ng nz pe
 cs=0d2f ss=004f ds=0aef gs=0be47 fs=0150b000    cr2=01550000  cr3=001d9000
doscall1:CONFORM16:postDOSSEMWAIT:
002d|d02f:0000272d c9 leave ;bc0
```

```
##bp d02f:272d,"j wo(tasknumber)==2d,'.r;r ss=0;g';g"
```

```
##g
eax=00000000 ebx=00000014 ecx=0009a9ef edx=0000a9ef esi=00090bff edi=00090006
eip=00002727 esp=0000b228 ebp=0009b230 iopl=2 -- -- nv up ei ng nz pe
 cs=0d2f ss=004f ds=0aef gs=0be47 fs=0150b000    cr2=01550000  cr3=001d9000
d02f:0000272e ca0800  retf 0008
```

```
Symbols linked (calc)
DelayHardError SYS3171: 4 string(s):
  Pid 000b  Tid 0001  Slot 002d  HobMte 03be
C:\OS2TOOLS\MRFILEPM.EXE
```

```
c0000005
1a05272d
P1=00000008 P2=6d640000 P3=XXXXXXXX P4=XXXXXXXX
EAX=00000000 EBX=0000014 ECX=0009a9ef EDX=0000a9ef
ESI=00090bff EDI=00090006
DS=0aef DSACC=00f3 DSLIM=00000fff
ES=be47 ESACC=00f3 ESLIM=000017f5
FS=150b FSACC=00f3 FSLLIM=00000030
GS=0000 GSACC=**** GSLIM=********
CS:EIP=d02f:0000272d CSACC=00d4 CSGlobal=00054a3
SS:ESP=0000:0000b230 SSACC=**** SSLIM=********
EBP=0009b230 FLG=00002386
```

```
DOSCALL1.DLL 0005:0000272d
```

This technique will successfully terminate an application. If `TRAPDUMP` is set appropriately then a system dump will be taken.

If `TRAPDUMP` is not correctly set for taking dumps, it may be dynamically modified from the debugging console. Symbol `DumpDevice` specifies the dump partition or drive letter (without the colon) and `DUMP_ON` is a flag byte that take values 0, 1 or 2 to specify whether `TRAPDUMP` is `OFF`, `ON` or `R0` respectively. Use the `E` command to modify these fields according to needs. For example, if we wish to set the equivalent of `TRAPDUMP R0,F` after system initialisation then the following command sequence would achieve this:

```
e dump_on 2
```

```
e dumpdevice "F"
```

When examining a dump taken by calling `RASRST`, directly or indirectly, using the `G` command then the registers at the time the Kernel Debugger was last entered can be found at label `_RegSA`. The format of this save area is as follows.

Before fix pack 29 for Warp 3.0 and base Warp 4.0:

```
Offset Register mnemonic
+0  EAX
+4  EBX
+8  ECX
+c  EDX
+10  ESP
+14  EBP
+18  ESI
+1c  EDI
```

```
Offset Register mnemonic
+0  EAX
+4  EBX
+8  ECX
+c  EDX
+10  ESP
+14  EBP
+18  ESI
+1c  EDI
```

```
```
+20  ES
+22  SS
+24  DS
+26  FS
+28  GS
+2a  EIP
+30  reserved
+34  EFLAGS
+38  MSW
+3c  GTD limit
+3e  GTD base
+42  reserved
+44  IDT limit
+46  IDT base
+4a  reserved
+4c  LDTR
+4e  TR
+50  CR2
+54  CR3
+58  DR0
+5c  DR1
+60  DR2
+64  DR3
+68  DR4
+6c  DR5
+70  DR6
+74  DR7
+78  reserved
+7c  TR6
+80  TR7

From fix pack 29 for Warp 3.0 and base Warp 4.0:

**Offset** **Register mnemonic**
+0  EAX
+4  EBX
+8  ECX
+4c  EDX
Virtual Memory Management Lock Trace
Virtual Memory Management implements a logging function that records successful attempts to lock and unlock memory pages.

Memory locking and unlocking is implemented by the Memory Management routines: `VMLockMem` and `VMUnlock`. This routine is available directly to all kernel components and indirectly to device drivers through:

- `DevHlp_Lock`
- `DevHlp_Unlock`
- `DevHlp_VMLock`
- `DevHlp_VMUnlock`

and to file system drivers through:

- `MFSH_Lock`
- `MFSH_Unlock`

The VM lock trace is activated by setting bit 0 of the VM log flag double-word to 1. The flag double word is located at symbol `_VMLogFlags`. Since no function is currently assigned to the other bit positions so the lock log may be effectively turned on by setting the byte a `_VMLogFlags` to `0xff` as in the following example:

```
e _vmlogflags
%ffff0127c 00.
```

The fields displayed in each lock trace entry are formatted from the constituent parts of the corresponding lock handle. They are defined as follows:

- **L**
  - Indicates a lock request.

- **U**
  - Indicates an unlock request.

- **base**
  - The virtual page number (that is the high order 5 digits of the address) of the page(s) to be locked or unlocked.

- **size**
  - The number of pages being locked or unlocked

- **flags**
  - The following bit settings are defined:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>Lock is a long-term</td>
</tr>
<tr>
<td>0x02</td>
<td>Verify lock call</td>
</tr>
<tr>
<td>0x04</td>
<td>Lock originated from a DevHlp</td>
</tr>
</tbody>
</table>
The **hob** of the memory object whose pages are being locked or unlocked.

The **hptda** of the process that requested the memory lock or unlock.

The **ret** is the return address from **VMLockMem**, that is, the address of the caller.

**Note:**

The return address is unfortunately of limited use since most calls to **VMLockMem** are made via a limited number of interface routines. In particular, **DevHlp** lock requests are made via **dhw_VMLock** and **SegLockDM**. Unless one can trace in addition the **SS:ESP** on entry to **VMLock**, the lock trace alone will be insufficient to solve memory locking problem. One possible way of providing more information is to supplement the lock trace with following breakpoint commands:

```
##bp _vmunlock+1,"k ss:sp;g"
##bp _vmlockmem+1,"k ss:sp;g"
```

Refer to the Kernel Debugger **K command** and **BP command** for further information.
Related information on memory locking may be found under the description of the Kernel Debugger .MO command.

The latest versions of OS/2 2.11 and OS/2 3.0 have implemented a new Kernel Debugger command that facilitates an alternative method for analysing memory locking problems. See the Kernel Debugger .MK command command for details.

--------------------------------------------

**Virtual Memory Management System Heap Validation**

The system will perform additional validation of the kernel heap structures under the debug kernel if the byte at label: _vmkhGflags is set to a non-zero value.

There is a noticeable performance overhead when this option is activated. Therefore it is recommended that it is only used when a heap corruption problem is suspected.

The system will validate the linkages between various heap structures. If an error is detected then an IPE is generated with one of the following messages:

- VMKSH: Invalid hint pointers
- VMKSH: Invalid number of ksh descriptors
- VMKSH: Invalid number of ksh blocks
- Invalid heap block header at addr: ssss:oooooooo
- Preceding block at addr: ssss:oooooooo
- No preceding block

--------------------------------------------

**System Loader Logging Facility**

The system loader provides optional logging and checking under the debug kernel. These optional facilities may be activated selectively by setting bits in the _LdrDebugFlags flags double-word as follows:

**Note:**

The flags described are those implemented in OS/2 Warp V3.0. Slightly different, similar messages are generated for earlier releases of OS/2.

**0x00000001**

This will cause the Loader to break into the debugger using an INT 3 instruction if any of the following error conditions are detected:

- Not enough memory
- Caching error
- Invalid Ordinal
- Procedure not found
- Bad EXE format
- Invalid segment number
- Invalid CALLGATE
- Network Disconnected

**0x00000002**

This will generate log entries when LDRGetPage exits with a non-zero return code. LDRGetPage is called to
demand load a page within a object of a load module. The message logged is of the following form:

```
1drGP bad cr2=nnnnnnnn rc=mmmmmmmm
```

cr2= is the page fault address and rc= is the LDRGetPage return code.

**0x00000004**

This generates log entries when LDRGetPage is called to demand load a page within a object of a load module. The message logged is of the form:

```
1drGP cr2=nnnnnnnn hMTE=hhhh bno=oo
    name=pppppppppppppppp
```

cr2= is the page fault address,

hMTE= is the module's hmte,

bno= is page number with in the module

name= is the module's full name taken from the SMTE.

**0x00000018**

This switch causes log information to be generated when DLL modules are loaded and initialised. The following messages are logged:

```
1drDLM entry - slot ssss ptda pppppppp
1drDLM name - slot ssss name nnnnnnn
1drDLM free - slot ssss
1drDLM exit - slot ssss
tk SD has-init slot=ssss
tk SD no-init slot=ssss
tk SD pre-inc slot=ssss cnest=nnnn
tk IN pre-dec slot=ssss cnest=nnnn
```

slot is the thread slot in which the DLL is being processed,

ptda is the address of the PTDA for this slot

name is the DLL module name

cnest Nesting counter for TKLibStartDispatch

1drDLM entry marks entry to w_loadmodule, the DosLoadModule worker routine.

1drDLM name marks the successful request for the DLL initialisation mutex semaphore (ptda_DLMsem (PTDA +0x4ac (H/R: +0x4a8))).

1drDLM free marks the release of the mutex semaphore. Exit marks the exiting of w_loadmodule.

1drDLM exit

tk SD marks events in TKLibStartDispatch.

tk IN and tk Lin Mark events in TKLinInitNextDLL

**0x00000080**

This switch requests import initialisation be recorded. Messages of the following format are generated:

```
lpi, Recording init hMTE=hhhh, flags1=ffffffff, name=nnnnnnnnnn
lpi, Skipping init hMTE=hhhh, flags1=ffffffff, name=nnnnnnnnnn
```
hMTE is the module handle
flags1 are the flags MTE flags field. (See the .LM command for details).
name is the full module name taken from the SMTE.
module is the full module name taken from the SMTE.

lpi, Recording init
Logs the processing of system DLL imports from the system DLL names table in EXE file loading.
lpi, Skipping init
Logs system DLL names not imported in EXE file loading.
lpi, Processing imports
Logs the processing of DLL initialisation as the result of imports being present in an EXE module.
Irm, Recording init
Logs imported DLL initialisation being recorded.
Irm, Skipping init
Logs imported DLLs skipping initialisation.

0x00000100
Logs when the loader cannot load an object at the compiler/linker designated base address. The message logged appears as:

    Cannot load nnnnnnnn at the requested base address

where nnnnnnnn is the module name.

0x00000800
Logs the processing of the DLL import tree. The following messages appear:

    lpi, Processing imports slot=ssss, module=nnnnnnnnn
    ldr walking tree hMTE=hhhh, name=nnnnnnnn
    ldr walking tree going down
    ldr walking tree going up

lpi, Processing imports marks the initiation of the process for slot ssss and module nnnnnnnn.
ldr walking tree hMTE=hhhh, name=nnnnnnnn marks the processing of an imported DLL, whose handle is hhhh and name is nnnnnnnn
ldr walking tree going up marks a backward progression through the import tree.
ldr walking tree going down marks a forward progression through the import tree.

Example Loader Log
The following is an example of a loader log where all logging options have been activated. This illustrates the loader activity recorded when the FAXWORK.EXE icon was clicked on:

`lpi Processing imports slot=0022, module=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree going down`
`ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMGPI.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSPL.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSPL.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=02be, name=H:\OS2\DLL\SPL1B.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSPL.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=029e, name=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree going down`
`ldr walking tree hMTE=0419, name=H:\OS2\DLL\HELPmgr.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree going down`
`ldr walking tree hMTE=02ac, name=H:\OS2\DLL\PMCTLS.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=02ac, name=H:\OS2\DLL\PMCTLS.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE`
`ldr walking tree going down`
`ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWIN.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=029d, name=H:\OS2\DLL\PMWIN.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=0281, name=H:\OS2\DLL\SEAMLESS.DLL`
`ldr walking tree going down`
`ldr walking tree hMTE=02b1, name=H:\OS2\DLL\PMV1OP.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=02ac, name=H:\OS2\DLL\SEAMLESS.DLL`
`ldr walking tree going up`
`ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWIN.DLL`
`ldr walking tree going down`
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going down
ldr walking tree hMTE=0362, name=H:\OS2\DLL\IBMS332.DLL
ldr walking tree going up
ldr walking tree hMTE=0362, flags1=2098b398, name=H:\OS2\DLL\IBMS332.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=0368, flags1=2098b388, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going up
lrm, Skipping init hMTE=0362, flags1=2098b398, name=H:\OS2\DLL\IBMS332.DLL
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\OS2\DLL\IBMS332.DLL
tk SD no-init slot=36
ldr walking tree hMTE=036a, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going up
ldr walking tree hMTE=037e, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going down
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going up
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=0282, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMGROUP.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going down
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going up
ldr walking tree hMTE=037e, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going down
ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL
ldr walking tree going up
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4c000 hMTE=5e1 bno=3d
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4b000 hMTE=5e1 bno=3c
name = H:\FAXWORKS\FAXWORKS.EXE
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name = H:\FAXWORKS\FX044.LOL
lp i Processing imports slot=0036, module=H:\FAXWORKS\FX044.LOL
ldr walking tree hMTE=060b, name = H:\FAXWORKS\FX044.LOL
ldr walking tree going up
lrm, Skipping init hMTE=060b, flags1=2098b1c8, name = H:\FAXWORKS\FX044.LOL
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name = SND
lp i Processing imports slot=0036, module=H:\MMOS2\DLL\SN D.DLL
ldr walking tree hMTE=00fe, name = H:\MMOS2\DLL\SN D.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name = H:\OS2\DLL\PMWIN.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name = H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name = H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name = H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name = H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name = H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name = H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name = H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=00fe, name = H:\MMOS2\DLL\SN D.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name = H:\OS2\DLL\PMWIN.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name = H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name = H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0104, name = H:\OS2\DLL\MSG.DLL
ldr walking tree going up
ldr walking tree hMTE=00fe, name = H:\MMOS2\DLL\SN D.DLL
ldr walking tree going up
lrm, Recording init hMTE=00fe, flags1=6098b396, name = H:\MMOS2\DLL\SN D.DLL
lrm, Skipping init hMTE=104, flags1=2098b388, name = H:\OS2\DLL\MSG.DLL
lrm, Skipping init hMTE=029a, flags1=2098b388, name = H:\OS2\DLL\PMWIN.DLL
lrm, Skipping init hMTE=029c, flags1=2098b388, name = H:\OS2\DLL\PMHAPI.DLL
lrm, Skipping init hMTE=0262, flags1=2098b388, name = H:\OS2\DLL\NL S.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13310000 hMTE=fe bno=12
name = H:\MMOS2\DLL\SN D.DLL
ldrGP cr2=13311000 hMTE=fe bno=13
name = H:\MMOS2\DLL\SN D.DLL
tk LIn slot=36 cnest=1
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name = PMCTLS
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=45000 hMTE=5e1 bno=36
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4e000 hMTE=5e1 bno=3f
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178f5000 hMTE=60b bno=6
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=e2000 hMTE=5e1 bno=51
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=178f6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL
1drGP cr2=178f5000 hMTE=60b bno=6
name = H:\FAXWORKS\FX044.LOL
1drGP cr2=80000 hMTE=5e1 bno=49
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=3c000 hMTE=5e1 bno=2d
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=39000 hMTE=5e1 bno=2a
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=fc000 hMTE=5e1 bno=68
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=fa000 hMTE=5e1 bno=69
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=40000 hMTE=5e1 bno=33
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=e5000 hMTE=5e1 bno=54
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=fb000 hMTE=5e1 bno=6a
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=2a000 hMTE=5e1 bno=1b
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=2c000 hMTE=5e1 bno=1d
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=77000 hMTE=5e1 bno=18
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=4d000 hMTE=5e1 bno=3e
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=55000 hMTE=5e1 bno=46
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=e0000 hMTE=5e1 bno=4f
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=10b000 hMTE=5e1 bno=7a
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=10c000 hMTE=5e1 bno=7b
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=f7000 hMTE=5e1 bno=66
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=1f000 hMTE=5e1 bno=10
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=20000 hMTE=5e1 bno=11
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=e0000 hMTE=5e1 bno=5e
name = H:\FAXWORKS\FAXWORKS.EXE
1drGP cr2=25000 hMTE=5e1 bno=16
name = H:\FAXWORKS\FAXWORKS.EXE
1drDLM entry - slot 36 ptda ab99a000
1drDLM name - slot 36 name H:\FAXWORKS\Fax.adp
lpi Processing imports slot=0036, module=H:\FAXWORKS\FAX.ADP
1dr walking tree hMTE=0618, name=H:\FAXWORKS\FAX.ADP
1dr walking tree going down
1dr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
1dr walking tree going down
1dr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
1dr walking tree going up
1dr walking tree hMTE=029a, name=H:\OS2\DLL\PMGRPI.DLL
1dr walking tree going up
1dr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
1dr walking tree going up
1dr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
1dr walking tree going up
1dr walking tree hMTE=01e5, name=H:\OS2\DLL\SESMGR.DLL
The following shows the loader sequence when FAXWORKS.EXE is terminated:

```
1drgP cr2=178c6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL

1drgP cr2=178c7000 hMTE=60b bno=8
name = H:\FAXWORKS\FX044.LOL

1drgP cr2=178c6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL

1drgP cr2=178c6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL

1drgP cr2=50000 hMTE=5e1 bno=41
name = H:\FAXWORKS\FAXWORKS.EXE

1drgP cr2=178e4000 hMTE=618 bno=15
name = H:\FAXWORKS\FAX.ADP
```

DosDebug Logging Facility
The kernel worker routines for the **DosDebug** API implement a number of logging functions for use in debugging errors in *DosDebug* itself. These are activated by setting bits in the double-word at symbol: `_DBGbugbug_`.

The following flags bits are defined:

- 0x01000000  
  Display input to *DosDebug*

- 0x02000000  
  Display output from *DosDebug*

- 0x00000010  
  Display exceptions in *DosDebug* processing.

- 0x10000000  
  Display execution flow in debugger processing.

- 0x20000000  
  Display execution flow in debuggee processing.

- 0x40000000  
  Display execution flow in watchpoint and debug register processing.

---

**DosPTrace Logging Facility**

The kernel worker routines for the **DosPTrace** API implement a number of logging functions for use in debugging errors in *DosPTrace* itself. These are activated by setting bits in the double-word at symbol: `_PTbugbug_`.

**Note:**

*DosPTrace* internally thunks to *DosDebug* therefore *DosDebug Logging Facility* may be a useful diagnostic aid with *DosPTrace*.

The following flags bits are defined:

- 0x01000000  
  Display output buffer set-up passed to user.

- 0x02000000  
  Display input buffer set-up passed from user.

- 0x04000000  
  Display conversion routine flow.

- 0x08000000  
  Display alias conversion routine flow.

- 0x10000000  
  Display input and output return codes only.

- 0x20000000  
  Display processing of notifications from *DosDebug*.

- 0x00000001  
  Display floating point information. processing.

---

**Kernel Debugger Breakpoints**

The **break-point command set** of the Kernel Debugger provides a mechanism for intercepting the execution of code through a particular
path. For debugging application programs, break-points are generally required within the application itself or on call to or return from one or more system APIs.

Each system API results either in a call to a system DLL or to the Kernel through a CallGate. The name of a system interface that is called when an application uses an API is either identical to the API name or may be determined from one of the following conventions:

- **DosI**\_name: Kernel Callgate name corresponding to API Dos\_name.
- **Dos32**\_name: DOSCALL1 32-bit entry point corresponding to API Dos\_name.
- **Dos16**\_name: DOSCALL1 16-bit entry point corresponding to API Dos\_name.

Other system DLLs such as PMWIN.DLL, PMMERGE.DLL, etc. adopt similar conventions, for example API **WinCreateWindow** calls **Win32CreateWindow** in PMMERGE.DLL.

In nearly all cases the system entry points have corresponding system tracepoints with the entry point name prefixed with either \_pre or \_post. Thus the **System Tracepoints Reference** provides a comprehensive source for deriving API related break-points.

Physical Device Driver helper routines pass through a common router, then to specific worker routines. Worker entry point names generally adhere to the following convention:

- **DosHlp**\_name: worker routine dh\_name.

Virtual Device Driver helper routines have entry points in the kernel with identical names (folded to uppercase) to the helper name.

File System Driver and Mini-File System Driver helper routines have entry points in the kernel with identical names to the helper name.

In addition to API and Driver Helper related break-points, the following system labels may also prove useful when intercepting errors or program initiation:

- **tkSchedNext**: This routine is called when a new thread is selected for scheduling. The out-going thread slot number is recorded in variable Tasknumber.

- **tkSchedNext** exits from one of two points:
  - **SchedNextRet**: A new thread slot is selected.
  - **SchedNextRet2**: The same thread slot is selected.

These labels maybe used to obtain a trace of dispatching activity. This is particularly useful when trying to establish the scope of hang conditions.

The following example illustrates how to obtain a trace of dispatched tasks using this break-point.

```plaintext
#bp _tkschednext,".p #;g"

Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
*003f# 0019 0000 0019 0001 blk 081e 7b98c000 7bb2d394 1bf8 10 wkstahlp
*003f# 0018 0000 0018 0002 blk 021f 7b98e000 7bb2d548 1f00 10 wksta
*003f# 0018 0000 0018 0002 blk 021f 7b98e000 7bb2d548 1f00 10 wksta
*003f# 0018 0000 0018 0003 blk 0200 7b996000 7bb2dc18 1eb8 10 wksta
*003f# 0018 0000 0018 0003 blk 0200 7b996000 7bb2dc18 1eb8 10 wksta
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
*003f# 0018 0000 0018 0001 blk 0500 7bb2a58 0eb8 01 pmshell
```

These labels may be used to obtain a trace of dispatching activity. This is particularly useful when trying to establish the scope of hang conditions.
Note:

The status shows as blocked since \_tkSchedNext has been called because the current thread is giving up its
time-slice.

DosLibDisp

This API is called to initiate DLL initialisation whenever a new module is loaded into memory. Since this is called for
every .EXE at load time, in the context of the new process and thread, it provides an excellent breakpoint for
intercepting the loading of a new module in a new process.

When DosLibDisp receives control, the MTE, SMTE have been created and the program module has been loaded.
From the SMTE we can determine the entry point of the new module and thus set a breakpoint on this address.

The following example illustrates how to set a breakpoint on entry to a new module.

```
>> Add breakpoint at DosLibDisp, then start CMD.EXE
##bp doslibDisp
##g
eax=00000000 ebx=000029f4 ecx=00000010 edx=00000014 esi=00000bc8 edi=00000c0a
eip=00000294 esp=0000773c ebp=00007752 iopl=0 -- -- nv up ei pl nz nc
c3=0 ds:0000 es:000f fs:0000 gs:0000 cr0=01000000 cr2=1fc70490 cr3=001d0000
doscall11:CODE|GROUP:DOSLIBIDISP:
ffd7:00000294 b80100 mov ax,0001 ;br0
#p#
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
*0044# 002c 0006 002c 0001 run 0400 7b9ae000 7bb4fc14 7bb2f088 1f48 19 cmd
>> The hmte for the current process is found in the PTDA at
>> ptda_module
##dw ptda_module l1
0030:0000ffaa 03a1
##lmo 3a1
hmte=03a1 pmte=%fe97ebe4 mflags=84903152 c:\os2\cmd.exe
obj   vsize    vbase    flags   ipagemap cpagemap hob  sel
0001 0000c6a8 00010000 80001025 00000001 00000002 03a0 000f r-x shr alias
0002 00007efa 00020000 80001025 0000000e 00000008 03a2 0017 r-x shr alias
0003 00009730 00030000 80001043 00000006 0000000e 0000 001f rw- prel alias
>> Now dump the MTE and SMTE, whose address is at MTE+0x4
##dd %fe97ebe4 l8
%fe97ebe4 03a10002 fd4341d0 fe97ec1c fe9a143c
%fe97ebf4 84903152 00000007 00060050 fe908e74
##dd %fd4341d0
%fd4341d0 00000000 00000001 00000002 00000003 00000004 00000005 00000006 00000007
%fd4341e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%fd434200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%fd434210 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%fd434220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%fd434230 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
%fd434240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
>> SMTE=0x4 is the entry point object number
>> SMTE=0x8 is the entry point offset offset
>> For CMD.EXE this is 2:44fa
>> Since object 2 starts at %20000, we can define a breakpoint on
>> entry to CMD.EXE at %20000+44fa
##bp %00020000 +44fa
##b1
0 e ffd7:00000294 [DOSLIBIDISP]
1 e %00244fa [\_astart]
>> Disable BP 0 since DosLibDisp is called for every DLL that will be
>> initialised in the new process.
##bd 0
VMLockMem

This breakpoint is on entry to the memory locking subroutine of Virtual Memory Management. It may be used in conjunction with the VM Lock Trace.

_XCPTBuildR3DispatcherStack

This routine is called whenever a process fatal exception is generated by the kernel, regardless of whether exception handlers are registered. It therefore makes a stronger method than VSF for intercepting fatal user exceptions.

Exception management and how to intercept exceptions is discussed in more detail in Trap and Exception Processing.

Dos32R3ExceptionDispatcher

This entry-point is DOSCALL1.DLL is called by the kernel to process all user exception handlers. A breakpoint on the label allows one to intercept user exceptions before the user context is modified by any user exception handlers. On entry ESP+0x4 contains the trap number, ESP+0x8 points to the exception report record and ESP+0xc points to the exception context record.

_xcptrR3ExceptionDispatcher.

This routine is called from Dos32R3ExceptionDispatcher to process each of the user exception handlers. It does this by locating exception registration records from the TIB at +0x0. On entry to the Ring 3 Exception Dispatcher, ESP+0x4 and EXP+0x8 point to the exception report record and exception context record, respectively.

The exception report record contains the exception number, and exception address. The exception context record contains all register values at the time of exception.

The layout for both these records is given in the BSEXCEPT.H header file of the OS/2 Programmer's Toolkit.

Most exceptions are generated from a hardware detected exception such as a trap. These are readily intercepted by using the Kernel Debugger VSF command. Exceptions may also be generated by the DosRaiseException API. Whatever the source all exceptions will eventually result in a call to _xcptrR3ExceptionDispatcher. This makes this label an excellent break-point for intercepting and filtering any exception that will drive a user's exception handler.

The following example illustrates the use of this break-point, where the system generates a C0000005 exception following a Trap E in an application program.

>> Break on entry to the Ring 3 Exception Handler Dispatcher
>> Intercept all fatal exceptions

Symbols linked (trape)

Trap 14 (0EH) - Page Fault 0004, Not Present, Read Access, User Mode
eax=00000000 ebx=00000000 ecx=0002059c edx=000a0000 esi=00000000 edi=00000000
eip=000022d8 esp=00000000 ebp=00000000 iopl=2 -- -- nv up ei pl nz na nc
cs=002b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=011d9000
doscall1:FLAT32:_xcptrR3ExceptionDispatcher:
005b:000022d8 8b00 mov eax, [eax] ;br0

>> %ESP+4 points to the exception report record
>> %ESP+8 points to the exception context record

##dd %esp
Dos32Exit and DosR3ExitAddr
Both these labels provide good breakpoints to catch an application terminating normally.

Dos32Exit is the entry point for the DosExit API. DosR3ExitAddr is the entry point in DOSCALL1.DLL, called when an application issues the return statement to return to the system.

Win32SetErrorInfo
This API is called by PM whenever it needs to record a PM error. When this is used as a break-point, the double-word at %esp+0x4 contains the PM Error code about to be recorded.

NWDHandler
This symbol is the entry point to the trap 2 interrupt handler. The IDT entry for trap 2 contains a Task Gate that points to NWDHandler. When NWDHandler receives control the Task Register will contain the selector for the current TSS. The link field of the current TSS will contain the previous value of the TR, where the processor saved the current registers when the interrupt occurred.

Frequently NMI interrupts are associated with disabled code and obscure hardware or software problems. If can be useful on these occasions to set up a KDB.INI file with the following commands to display information when the trap 2 occurs. This is particularly advantageous when dealing with NMI interrupts caused by the NMI Watch Dog timer firing.

```
bp nwdhandler,"? 'curr tss';dt tr:0;? 'prev tss';dt #(wo(tr:0)):0"
```
Note:

When the first NMI occurs, the following would be displayed:

curr tss

eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
eip=fff4074c esp=00000400 ebp=00000000 iopl=0 -- -- nv up di pl nz na po nc
cs=0170 ss=lea0 ds=0168 es=0168 fs=0000 gs=0000 cr3=001dd000
ss=0000 esp=00000000 ss=0000 esp=00000000 esi=00000000 esp2=00000000
prev tss

eax=00002ff ebx=139b0000 ecx=00000400 edx=00009ae8 esi=139b993c edi=139d0400
eip=1b7228fe esp=0006eeaa ebp=0006eee0 iopl=2 -- -- nv up ei pl nz na po nc
cs=005a ss=004a ds=0053 ss=0053 fs=150b gs=0000 cr3=001dd000
ss=0030 esp=00006d80 ss=0000 esp=00000000 ss=0056 esp=000f000
ports trapped: 0-fff

The register values when the NMI occurred are displayed under the label \textit{prev tss}.

After NDWHandler has processed the NMI it performs a task-switch back to the previous TSS, but only after editing the previous TSS to ensure that control is passed to TRAPCommonFaultEntry. The task switch is effected using IRETD with the NT flag set in EFLAGS. This leaves the NMI TSS' EIP pointing at the instruction following the IRETD at approximately NDWHandler+25. To allow more that one NMI to be handled the instruction following the IRETD is a JMP NDWHandler. Therefore whenever the NMI TSS' EIP doesn't point to the NDWHandler entry point it is a sure indication that at least one NMI has occurred.

--------------------------------------------

Trap and Exception Processing

The fine detail of exception management by OS/2 is complex. However the principles are easy to grasp. This section gives an overview of OS/2 Exception Management sufficient to provide the reader with a technique for intercepting exceptions in user code under the Kernel Debugger.

Exception Definition

Exceptions may be summarised as follows:

- Exceptions refer either to:
  - Hardware Traps and Faults - INTEL defined.
  - Software generated exceptions - OS/2 and User defined.

- Each Hardware Exception has an associated vector, which the processor uses to index the IDT to give control to the appropriate system exception handler.

- OS/2 Converts Traps and Faults to software exceptions. For example, traps 0xd and 0xe are converted to exception 0xc0000005.

- Software exceptions are generated from three sources:
  1. Converted Hardware Traps and Faults.
  2. Software Signals.
  3. Software Exceptions from DosRaiseException.

- Exceptions occur for both normal and abnormal reasons. In the normal case additional processing is required to be executed in a manner transparent to the main line code. Examples of this are:

  Page fault exceptions.
Trap 1 and 3 for system trace

387 Co-processor emulation

VDM privileged instruction emulation

In the abnormal case, an error condition has been detected. If the error cannot be corrected then either a process or the system dies depending on whether the error can be isolated to a particular process. Usually traps and faults in ring 0 code result in system termination. Bad parameters passed in system APIs may cause the kernel to trap. The system recovers by directing an exception 0xc0000005 to a process. Unless the process can handle this exception, it dies.

- Full details of OS/2 defined exceptions are given in OS/2 System Exception Codes.

Exception Logic

The essential logic for exception handling is as follows:

- If the processor generates a hardware exception then control is given to the first level exception handler pointed to by the IDT descriptor that corresponds to the hardware exception vector.

- If the Kernel Debugger Vector Commands have been specified without the fatal flag then first level exception handlers have been replaced by the Kernel Debugger routines. These may give control to the debugging console or enter the normal system handlers if interception criteria are not satisfied.

- The non-debugger first level routines perform any specific processing for the current exception, for example processing single step and breakpoint traps.

- If full recovery is possible then the first level routines exit with an IRET instruction.

- In most cases control passes from the first level trap handlers to TrapCommonFaultEntry. This performs common processing for all hardware exceptions. If recovery is possible, for example by satisfying a page fault or making a segment present, then this is done and control returned to the interrupted code.

If recovery is not directly possible or further special processing is required then control passes to one of the following second level exception handlers:

- V8086 Emulation for instruction emulation.

- VDM Exception Handler to reflect non-fatal exceptions back to the VDM using its IDT.

- Process Fatal Fault Handler (_TRAPProcessFatalFault) for non-kernel mode code (InDos=0).

- Kernel Fault Handler for kernel code (InDos=1)

Special handlers for Co-processor handling, NMIs etc..

- The Kernel Fault Handler checks for the presence of a local fault handler by inspecting TSDpnFault. If this is non-zero then passes control to the local fault handler, otherwise it passes to the System FatalFault handler (SystemFatalFault).

- The System Fatal Fault handler will enter the Kernel Debugger (if in a non-RETAIL kernel), otherwise it will call and Device Drivers that have registered for notification of fatal system faults, then exit to the panic routine with a formatted message - usually the IPE trap screen. Once in panic the system will not dispatch any more threads. If TRAPDUMP or REIPL are specified then these are acted on otherwise the system waits to be re-booted.

- The Process Fatal Fault handler will check for fatal fault interception by the Kernel Debugger (VSF command) and enter the kernel debugger if interception criteria are satisfied. Otherwise user exception processing begins, if it is not possible to dispatch user exception handlers then DelayHardErr is called immediately to build the trap screen and wake the Hard Error process.

Normally control passes to the _XCPTBuildR3DispatcherStack.

- _XCPTBuildR3DispatcherStack is entry-point for all kernel initiated exceptions to be sent to user. It is responsible for massaging the users stack so that when the kernel exits, control returns to the Exception Dispatcher (Dos32R3ExceptionDispatcher in DOSCALL1.DLL).

The parameters to XCPTBuildR3DispatcherStack are:

- Trap number or 0x0000ffff for S/W generated exceptions

- Exception number

- Count of exception info parameters

- Pointer to the array of exception info parameters

- Boolean, if true then exception is non-continueable.

- Pointer to any additional nested report record
The parameters to `Dos32R3ExceptionDispatcher` are:

- Trap number
- Pointer to the exception report record
- Pointer to the exception context record

If no exception registration records exist for the current thread then the thread enters termination and the Exception Dispatcher is not called.

- The Exception Dispatcher runs the chain of exception registration records, anchored from the TIB of the current thread. Each registered user exception handler is called in turn (via an intermediate routine, `_xcptExecuteUserExceptionHandler`). The return code (exception disposition) passed back by the exception handler is examined. If it specifies `XCPT_CONTINUE_EXECUTION` then control returns to the kernel via `Dos32ExceptionCallBack`, whereupon the thread's stack is prepared for returning to the interrupted program. If `XCPT_CONTINUE_SEARCH` is specified then the next exception handler in the chain is dispatched. When the last exception handler has been dispatched (and all have returned `XCPT_CONTINUE_SEARCH`) then control passes to the kernel via `Dos32ExceptionCallBack` and the thread is terminated.

- `XCPT_CONTINUE_STOP` may be returned by a debugger via `DosDebug` to indicate that debugger handler the exception and that exception handler scheduling should be halted immediately.

The values for the various return codes are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XCPT_CONTINUE_SEARCH</td>
<td>0x00000000</td>
<td>Exception not handled</td>
</tr>
<tr>
<td>XCPT_CONTINUE_EXECUTION</td>
<td>0xFFFFFFFF</td>
<td>Exception handled</td>
</tr>
<tr>
<td>XCPT_CONTINUE_STOP</td>
<td>0x00716668</td>
<td>Exception handled by debugger (via DosDebug)</td>
</tr>
</tbody>
</table>

The following additional return codes are used by internal exception handlers to manage nested exceptions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NESTED</td>
<td>0xf0f0f0f0</td>
<td>An exception occurred while an exception was active</td>
</tr>
<tr>
<td>COLLIDED_UNWIND</td>
<td>0x0f0f0f0f</td>
<td>Indicates collided unwinds</td>
</tr>
<tr>
<td>EXIT_UNWIND</td>
<td>0x65796C4B</td>
<td>Indicates the end of an exit unwind</td>
</tr>
</tbody>
</table>

- `Dos32ExceptionCallBack` is the kernel entry-point that is called after all user exception handlers have been called. It is passed the following parameters:

  - Trap number
  - Pointer to the exception report record
  - Pointer to the exception context record
  - Exception disposition

  This entry-point calls `_xcptExceptionCallBack` which either takes the default action according to the exception disposition and the exception type (fatal or non-fatal). The either user's context is retored from the exception context record or control passes to `_xctDefaultAction`. The latter action usually implies proces termination but also may result in a process or system dump being initiated.

  For fatal exceptions, `_xctDefaultAction` calls `DelayHardError` to format the trap screen information from the context record and wake the Hard Error process. It also writes the POPUPLOG.OS2 entry and calls the Kernel Debugger if the VSU command has been specified.

- Local Fault Handlers are exception handlers registered by kernel routines. Typically one is registered on entry to the kernel by an API call, and de-registered on exit. If a Local Fault Handler cannot resolve the fault then it will call panic if a serious system fault has occurred, or `_XCPTBuildR3DispatcherStack` if user code is at fault, for example when a bad parameter supplied to an API by an application program causes the kernel to trap.

- Multiple nested Local Exception Handlers may be registered. When the system calls a Local Exception Handler, the current handler deregisters itself and reinstates the previous nested handler. This is done by restoring the previous handler address from the top-most long-jump buffer saved in `TSDpljmp` and updating `TSDpmFault`. The register values saved in the long-jump buffer are restore then then fault handler returns to the main-line code that registered. EAX is set by the fault handler to indicate that an
error occurred. When no more local exception handlers are registed then TSDpfnFault is zeroed, thereby de-registering all local exception handlers. Local Exception Handlers are always deregistered on exiting the kernel.

- The DosRaiseException API is called to create a user exception. This passes control to _xcptExceptionCallBack and normal user exception processing follows.

These details are summarised in the following diagrams:

Exception Registration Records
OS/2 Exception Exception Management - Overview
Exception Handler Stack Frames

--------------------------------------------

Exception Registration Records
OS/2 Exception Exception Management - Overview
Exception Handling - Overview

Interrupt Descriptor Table

Specific 1st Level Trap Handlers Are Entered
(trap0, trap1, ...)

> IsET
  Fault Handled?

TrapCommonFaultEntry

R06 emulation

VDK

Process Fault

Kernel Fault

Enter Debugger
for VDF/MTF

Call Delay/Handler

Asynchronous Notification

_xcptExceptionHandler

Kernel Mode

User Mode

Jos32ExceptionDispatcher

Hardware
Process
(Exception Handler
(Display Trap Screen)

Local Fault Handler

LDpERMFault?

Enter Debugger

Call DD SPP Exits

Rerun Panic (TPR)

Continue

_xcptExceptionCallback

Terminate

Jos32ExceptionCallback
Exception Handler Stack Frames
Exception Handler Stack Frames

low address

- Exception handler ESP
- EIH
- pRePlac
- pRegAc
- pCtxAc
- 45 bytes of MDAIs

high address

- _except0.ExceptionDispatchISR
- _except1.ExceptionDispatchISR
- 35 bytes of MDAIs

This exception frame is repeated for nested No exceptions

- EIH
- 0
- Trap number
- pRePlac
- pCtxAc
- Report Record
- Context Record
- 128 bytes of MDAIs

TCB

TIB

Exception Reg Bec

Exception Reg Bec
Intercepting Exceptions and Traps

The following list provides guidelines for intercepting traps and exceptions under the Kernel Debugger for various circumstances:

Fatal exceptions occurring in application ring 2/3 code.

**BP _XCPTBuildR3DispatcherStack** will trap every software and hardware exception. The breakpoint is in the kernel, so use .R to display the registers at the time of the exception. This break-point works regardless of whether exception handlers are registered.

**Note:** If the exception is generated through use of an API (bad parameter or DosRaiseException) then the **CS:EIP** will point after the call gate instruction.

Fatal Hardware Traps and Faults in application ring 2/3 code.

**VSF * will intercept all such exceptions at the point of the exception.**

Fatal Hardware Traps and Faults in ring 0 code.

**VTF * will intercept all such exceptions at the point of the exception, providing no Local Fault Handler has been registered.**

All ring 0-3 traps and faults.

**VT * will intercept them all.**

All application ring 2/3 code traps and faults.

**VS * will intercept them all.**

Exceptions in application ring 2/3 code that will drive exception handlers.

**BP _xcptr3ExceptionDispatcher** will be intercepted if any are registered, but this will be called once to process the entire chain.

Each User Exception Handler.

**BP _xcptExecuteUserExceptionHandler** will be called to dispatch each exception handler. Alternatively use the registration records from the TIB to locate the entry point of a given exception handler.

Post User Exception Handling

**VSU** will intercept and unrecovered fatal exception delivered to ring 2 and 3 after exception handler processing.

**Note:**

User exception handlers can be disabled under the Kernel Debugger by locating the TIB, then storing 0xffffffff at offset 0x0, which is the pointer to the exception registration record chain. The chain is terminated by 0xffffffff and can be re-worked manually for debugging purposes - provided that the system is not already processing an exception for this thread.

Dump Formatter User Guide

The Dump Formatter is an interactive line-mode utility that supports a variety of commands for extracting and displaying information from a system dump. There are two versions of the Dump Formatter:

**df_ret.exe** The Dump Formatter for dumps from systems running either the RETAIL or HSTRICT kernels.

**df_deb.exe** The Dump Formatter for dumps from systems running the ALLSTRICT kernel.

**Note:** Refer to the Kernel Debugger User Guide for a discussion on the different OS/2 Kernels.

Each of the two Dump Formatters is generated for each build of the OS/2 kernel. Thus the Dump Formatter is system level and fix-pack level dependent, in a similar way to the debug kernels. Several base versions of the Dump Formatters are distributed with the OS2PDP package. Versions of the Dump Formatter that apply to a particular fix-pack may be obtained from the following sources:
• The OS/2 Base Product CDROM for WARP is distributed with the ALLSTRICT kernel and Dump Formatter. (For the initial release of WARP this was only available on the US version of WARP).

• The Developer Connection CDROM - this may be ordered through the Developer Assistance Program (DAP) or the System Library Subscription Service (SLSS).

• From your local IBM Marketing Representative.

• Calling the SDM BBS at USA 407-443-8000.

• For IBM internal users by FTP to the SDM at node:

  sdm.bocaraton.ibm.com
  or
  9.83.12.237

  Logon using Id and Password Anonymous. A list of files is contained in files.bbs.

The Dump Formatter has a named pipe interface that allow it to be controlled from another program. This is exploited by the PMDF program, which is also distributed with the OS2PDP package.

PMDF provides:

• A PM interface to the Dump Formatter.

• Automatic Dump Formatter version management.

• The ability to log output to a file.

• Use of Drag and Drop on Dump Formatter output to the PMDF commands line.

• A REXX interface that allows REXX EXECs to issue Dump Formatter commands and capture their output.

• Process Dump Formatting.

The command set supported by Dump Formatter is very similar to that of the Kernel Debugger. In many cases they share common commands. These are documented in the Kernel Debugger and Dump Formatter Command Reference.

Taking a System Dump

A system dump may created by any of the following means:

• Manually by using the command sequence:

  Ctrl-Alt-F10-F10 or Ctrl-Alt-Numlock-Numlock

• Automatically when an application or system trap occurs. See the description of the TRAPDUMP CONFIG.SYS setting for details.

• Directly from an application program by using the DosForceSystemDump API.

• Directly from a physical device driver by calling the VectorSDF address returned by the DevHlp_GetDOSVar helper service.

• Under the kernel debugger from the debug console by giving control to the RASRST routine. See Forcing a System Dump From the Kernel Debugger for further details.

Note:

Whilst it can be advantageous to take a dump when the system trace is active, it is not a pre-requisite. Occasionally system trace has been found to caused other traps.

Dump Formatter Installation
The Dump Formatter may be installed together with PMDF by using the installation procedure supplied with the OS2PDP package. Alternatively copy the *.EXE files to either a private directory or a directory in your current PATH. The only files the Dump Formatter accesses implicitly are Symbol Files, which if used, are convenient to have installed in the same directory as the *.EXE program files.

The command line syntax for Dump Formatter is as follows:

```
DF_RET     dumpfile                                     `-P pipname
DF_DEB
```

The parameters have the following meaning:

- **dumpfile**: The file name of the (decompressed) dump to be analysed. If a path is not prefixed to the file name then the Dump Formatter assumes the current path. See Dump Decompression below.

- **-P pipname**: The name of a named pipe through which Dump Formatter output and commands are channelled.

  **Note**: This parameter is intended for use when df_ret.exe or df_deb.exe is started from another program using the DosExecPgm API.

**Note:**

If no parameters are entered then the Dump Formatter give a syntax message. This message implies that a COM port may also be used as an interface, but this has not been implemented.

When the Dump Formatter is started it displays the build level of the system from which the dump was taken and then the build level of the formatter. If these do not match unpredictable results may occur. However, if the levels are close then it is probably safe to use the Dump Formatter, though not guaranteed.

If the incorrect type of Dump Formatter is used, for example retail Dump Formatter with a ALLSTRICT dump, then the Dump Formatter will probably trap. If it does not, then an error message will appear.

In general the dump formatter traps for one of three reasons:

- The reasons stated above, where there are type and level mismatches.
- The dump file is incomplete or corrupted.
- The Dump Formatter stack overflows.

The latter problem usually occurs when the _P command_ is used. This is sometimes circumvented by using the EXEDHR utility to increase the stack size of the Dump Formatter. Another approach is to use the %PS REXX to display each thread slot individually.

As part of the initialisation sequence, the Dump Formatter attempts to load symbol files, from the current directory, for each module that was loaded on the dumped system.

**Notes:**

Windows, WINOS2 and DOS symbol files are not usable under the Dump Formatter. However, the SYMLST REXX exec in the tools directory of the accompanying CD-ROM may be used to list a symbol file. This can sometimes be used in conjunction with the Dump Formatter provided that at least one location of a module or its data can be determined absolutely.

Symbol files not present in the current directory may be manually loaded using the WA command. The syntax and function of this command differs subtly from the Kernel Debugger equivalent:

- Under Dump Formatter names are symbol file names unlike Kernel Debugger where they are symbol map names. This allows relative path names to be used.
- Under Dump Formatter WA reads the symbol file, whereas under Kernel Debugger it is just marked active provided it was loaded when the module was loaded.

The Dump Formatter prompts for command input with a single # sign. Unlike the Kernel Debugger this is not used to signify the processor mode or whether paging is enabled. Consequently the Dump Formatter always assumes that the current processor mode is Protect Mode.
with Paging Enabled. The user must therefore explicitly prefix segment:offset addresses in Virtual 8089 mode with an ampersand (&).

Commands may be interrupted by pressing the ESC key.

Dump Decompression

Dumps may be taken either to a dedicated FAT hard disk partition or to diskette. For details on setting up the dump partition refer to the TRAPDUMP CONFIG.SYS command description.

Dumps taken to a hard disk partition may be used directly by Dump Formatter or PMDF.

Dumps taken to diskette have their data compressed and have to be decompressed to produce a single dump file. This may be done from within PMDF by selecting the New option of the File pull-down. PMDF offers the additional facility of decompressing diskette dumps directly from diskette images created by OS2IMAGE. See PMDF File Menu below, for details.

Sometimes PMDF fails to decompress a dump. This normally occurs when diskette 1 has not been re-inserted to complete the dump process. If this happenend the PATCHDMP utility may be used to correct the dump header on the first diskette. PATCHDMP may be found on the accompanying CDROM.

If PMDF is not being used then the DCOMP command may be used to decompress a dump. The syntax for DCOMP is as follows:

```
NDCOMP source drive file name /f
```

/source drive

Undocumented.

Specifies the drive where the DUMPDATA.nnn file will be found. This may specify either a hard disk drive or a diskette drive. The DUMPDATA.nnn files from a diskette dump may be copied to a hard drive root directory before using NDCOMP.

/file name

The target dump file name including path information.

PMDF Installation

PMDF provides a convenient front-end to the Dump Formatter. By installing PMDF in an appropriate directory structure it is able to select automatically the correct version of Dump Formatter for the dump to be analysed.

PMDF is installed by using the installation of the OS2PDP package. The resulting directory structure is as follows:

```
\PMDF\      PMDF.EXE PMDF.INF PMDF.HLP PMDFMSG.DLL PMDFVERS.LST *.CMD
\GA21\      DF_RET.EXE DF_DEB.EXE *.SYM *.SDF
\GA21MR1\   DF_RET.EXE DF_DEB.EXE *.SYM *.SDF
\Warp\      DF_RET.EXE DF_DEB.EXE *.SYM *.SDF
\Warp_fp\   DF_RET.EXE DF_DEB.EXE *.SYM *.SDF
             .  .  .  .  .
```
Each version of OS/2 is represented by a subdirectory containing the Dump Formatters and symbol files and structure definition files for that version.

The home directory contains the PMDF executables and help files, the version control file - PMDFVERS.LST, and any REXX EXECs to be installed in their default directory.

**PMDFVERS.LST**

More versions of the Dump Formatter may be installed by creating a new subdirectory for the new Dump Formatter and adding an entry to the PMDFVERS.LST file. Each entry of this file corresponds to an OS/2 build level or version. The format of an entry is as follows:

relative path:build level:descriptive text

**Notes:**

The path is relative to the home directory.

The build level is the internal system build level and may be determined either by browsing the OS2KRNL load module and searching for the text @#IBM:n.nnn#@ near the end of the module, or by using the BLDLEVEL utility in the OS2 directory. The VER /R command is not reliable since it only reports the base version level, not the fix-pack version level, in some releases.

The directory structure in the example above would be represented by the following entries:

- ga21:6.514:OS/2 2.1 General Availability
- ga21mr1:6.617:OS/2 2.11 MR1
- warp:8.162:Warp
- warp_fp:8.200:Warp Full Pack

--------------------------------------------

**PMDF Menus and Options**

PMDF offers a number of facilities from its pull-down menus and also from the mouse buttons.

From the Keyboard **Ctrl-C** and **Esc** serve to interrupt the Dump Formatter.

**Warning:**

Do not use the Dump Formatter **Q command**. Under PMDF this may cause PMDF to hang. To terminate the Dump Formatter either quit PMDF from the system menu or select another dump for processing.

The PMDF screen appears as follows:
PMDF File Menu

The File pull-down menu offers the following options:

New Dump

Select this option to decompress a new dump.

Notes:

For diskette dumps the DUMPDATA.nnn files may be copied for a directory on the hard drive and decompressed from there.

PMDF has the ability to decompress diskette images created by OS2IMAGE without re-creating the original diskettes. To use this facility each of the image file must be named image.nnn where nnn is a numeric sequence number that corresponds to the disk number.

Open Dump

This option prompts the user for the dump file name then invokes Dump Formatter.

Log Output

This option prompts the user to start or stop logging output to a file. Data may be appended to an existing log file.

Save Output

This option allows the user to save all output displayed in the PMDF scrollable window.

Connect

Connect allows PMDF to be used as a terminal emulator to drive a Kernel Debugger session. See the Kernel Debugger User Guide for more information.

Disconnect

Disconnect terminated the communications session with the Kernel Debugger.

The following diagram illustrates the File pull-down menu options.
PMDF Edit Menu

The Edit pull-down menu offers the following options:

Search String

Locates text within the scrollable window.

Undo

Reverse the previous Edit Cut action.

Copy

Copy marked text to the clip board.

Cut

Move marked text to the clip board.

Clear Screen

Clears the scrollable window of all text. This is not a reversible action.

The following diagram illustrates the Edit pull-down menu options.
PMDF Options Menu

The Options pull-down menu offers the following options:

Font Settings
This allows font selection for displayed output.

Function Keys
This provides a menu to redefine function keys as strings of Dump Formatter command strings. Commands may be separated by a semi-colon.

Terminal Settings
Allows the communications parameters to be specified for when the Connect option of the File pull-down is selected.

Save Settings
This will save the current options in PMDF.INI for use next time PMDF is started.

The following diagram illustrates the Options pull-down menu options.
The Analyse pull-down menu offers four selections, each of which displays its own menu selection. Where parameters are required they should be highlighted by double-clicking mouse button 1 on text in the scrollable window.

CAUTION:

The output from the Analyze options needs to interpreted with care. Some options are precise since they follow control block chains anchored from the SAS, for example the Physical Device Driver Chain and Kernel Heap. Others depend, for correct results, on correct symbols being loaded. Some options, for example those that display stacks, are more speculative in what they display.

Before these facilities are relied on, the user should thoroughly acquaint themselves with the manual techniques that belie their function. This information is available in the course materials that comprise the first section of this Handbook.

The following selections are available:

System

The System menu display the following options:
The Process menu display the following options:

- System
- Thread
- Synopsis
- Physical Device Driver
- Virtual Device Driver
- Interrupt Stack
- Program List
- Window Info
- Open Files
- Heap Info
- Memory
- Trace
- Process Info
- Thread Chain
- Module Table
- Local Descriptors
- Memory Objects
- NOS

The Threads menus dumps stacks related to a given thread. The following menu is displayed:
Synopsis

This offers a miscellaneous collection of options, the most important of which is the Trap Screen display. The following menu is displayed:

PMDF Help Menu

The Help pull-down menu offers standard help facilities.

The following diagram illustrates the Options pull-down menu options.
PMDF Mouse Options

Standard CUA mouse selection and highlighting are implemented. Marked items may be dragged and dropped onto the command line.

A double-click with mouse button 1 will highlight a blank delimited string.

A single click with mouse button 2 will display pop-up menu whose items take the highlighted text in the scrollable output window as input.

The following diagram shows an example of the mouse pop-up menu. In this example the Structures option is displayed. This particular option acts as a supplement to the Dump Formatter .D command. For it to work correctly, the Structure Definition Files (*.SDF) are required to be present in the same directory as the Dump Formatter. These files are build level dependent and will only display correct information if matched to the dump level. There is no validation performed by these displays. The user must ensure that an appropriate input address is highlighted.
PMDF REXX interface

PMDF provides a REXX interface that allows REXX EXECs to issue Dump Formatter commands and capture their output in REXX variables. EXECs are able to display output on PMDF's scrollable output window, command line and enter.

EXECs are invoked by entering the REXX EXEC name, with optional directory information, prefixed with a '%' character from the PMDF command window. If the exec is not installed in a directory in the PATH or in the same directory as PMDF then it must be prefixed with the fully qualified path name. For example:

%%%SEGTAB 123
%C:\MYEXECS\TEST1 parm1 parm2

It is also possible to use relative path expressions thus:

%..\SEGTAB 123

If a path has to be specified when passing an exec name as a parameter to another exec then quotation marks around the path and file name will be required.

PMDF implements its interface to the Dump Formatter by creating a REXX subcommand environment. The REXX address instruction allows an EXEC to execute and capture the output from a Dump Formatter command by addressing this subcommand environment.

The syntax and parameters for this implementation of the address instruction are:

address df 'CMD' <output> <df_cmd>
Where:

<output> is the name of a stem to a REXX compound variable that will be assigned to capture output from the Dump Formatter command.

"output.0" will be set to the number of lines. "output.n" will contain the nth line of output.

<df_cmd> is the dump formatter command and parameters.

Parameters following the EXEC name will be passed to the EXEC as a one parameter string.

A number of general purpose EXECs are provided in the OS2PDP package on the CD-ROM accompanying this book. These are:

RUNCHAIN Generalised Control Chain Running EXEC.
PS Generalised EXEC for executing Dump Formatter commands per thread slot.
TEMPLATE A Template EXEC containing a collection of subroutines useful for writing other EXECs.

There are also a number of example EXECs that format control blocks and illustrate how to use the REXX interface and the subroutines contained in TEMPLATE.

--------------------------------------------

The RUNCHAIN EXEC

Syntax

RUNCHAIN <addr> link(<offset>,<s>) stopvalue(<stop>) chain(<nnn>) exec(<cmd>)

print(<file>)

This exec provides a generalised control block chaining facility, where at each hop of the chain a command or exec may be executed. The starting address and link offset are required. Other parameters are optional. The parameters to RUNCHAIN are:

<addr> is an address expression of the start of the chain

<offset> specifies the decimal or hexadecimal offset of the linking address. Default is 0

<s> specifies the length of the linking field as: D (double) or W (word) - Default is D

<stop> specifies a termination value for the linking field. This take precedence over <chain> and may be specified as a hexadecimal or decimal value.

<nnn> specifies the maximum number of chain hops to traverse. Default is 10

<cmd> specified a command to be executed at each hop. If the command is prefixed with a % then an exec is executed. @L will cause the linear address of the current block to be substituted. Default is DD @L L4.

<file> specifies a print file to which the output will be copied.

Note:

Hexadecimal values are specified as 'nnx'

As an example: suppose the linear address of an MTE is %fff2bde0. MTEs are linked at +c in os2 2.1. To run the chain of MTEs displaying 8 double words do the following:

%RUNCHAIN %fff2bde0 link(c) exec(DB @L L40)

The resulting output would be appear thus:

Block 1 at %FF2BDE0
To format the first 40 MTEs in the chain do:

```
%RUNCHAIN %fff2bde0 link(c) exec(.lmo @L) chain(40)
```

The PS EXEC

**Syntax**

```
PS <s1> <s2> <cmd> <parms> <;cmd> <parms> ..... 
```

This is the Per-Slot exec. It will repeatedly execute a DF command string or REXX exec for each thread slot in the range specified. The linear addresses of slot related control blocks (TCB, PTDA and TSD) may be specified symbolically in the command string so that the correct address will be substituted for each slot traversed by PS.

The parameters to PS are:

<s1> Starting (hexadecimal) slot number

<s2> Ending (hexadecimal) slot number or *, which signifies highest active slot in the system.

<cmd> is any string of DF commands separated by ; or a single REXX exec prefixed by %.

<parms> are any valid parameters where @TCB, @PTDA and @TSD are substituted with their corresponding linear addresses. @disp is the scheduler's ESP relative to the TSD. N.B @disp is only defined when page table entries are present for the TSD.

**Example 1:**

Display priority information (on a 2.11 system) for slots 30 to 33 where priority class is at TCB+e4, priority delta is at TCB+e5 and dispatching priority is a word at TCB+e8.

Enter:

```
%PS 30 33 DB @TCB+e4 L2; DW @TCB+e8 L1
```
Slot 30
Warning: not all addresses are present
DB %7BA8FE78+E4 L2; DW %7BA8FE78+E8 L1
%7ba8ff5c 02 0f ..
%7ba8ff60 020f

Slot 31
DB %7BA9002C+E4 L2; DW %7BA9002C+E8 L1
%7ba90110 02 00 ..
%7ba90114 0200

Slot 32
Warning: not all addresses are present
DB %7BA9002C+E4 L2; DW %7BA9002C+E8 L1
%7ba90110 02 00 ..
%7ba90114 0200

Slot 33
DB %7BA90394+E4 L2; DW %7BA90394+E8 L1
%7ba90478 03 00 ..
%7ba9047c 0000
ps ended rc: 0

Note:
For slot 30 a warning message is issued because in this instance .s30 gave an error because slot 30 page tables were swapped out.

--------------------------------------------

The TEMPLATE EXEC

Template is not intended to be executed. Rather, it is a model for creating new execs. It contains a number of generally useful subroutines used in other execs.

Currently included in TEMPLATE are the following subroutines:

linaddr <address>
Converts an address expression to a linear address (without the % prefix). If storage cannot be referenced then a null string is returned.

getstor <h>,<a>,<s>,<f>
Retrieve a byte, word or double word from storage. If storage can’t be retrieved then the DF error msg is returned.

<
<h> is a dump handle
<a> is a DF address expression
<s> is the size specified as: B, W or D
<f> is the optional output format, which may be specified as C for character, N for decimal or X for hexadecimal string. X is the default.

gethxstr <h>,<a>,<l>
Retrieve a string of hex bytes from storage. If storage can’t be retrieved then a null string is returned. The string is returned as a concatenated string of bytes.

<h> is a dump handle
<a> is a DF address expression
<l> is the length of storage to retrieve
getbytes <h>,<a>,<l>
Retrieve a one or more bytes from storage. If storage can’t be retrieved then a null string is returned. The string is returned as a string of bytes separated by blanks.

<h> is a dump handle
<a> is a DF address expression
<l> is the length of storage to retrieve

getwords <h>,<a>,<l>
Retrieve a one or more words from storage. If storage can’t be retrieved then a null string is returned.

<h> is a dump handle
<a> is a DF address expression
<l> is the length of storage to retrieve

getdwords <h>,<a>,<l>
Retrieve a one or more double words from storage. If storage can’t be retrieved then a null string is returned.

<h> is a dump handle
<a> is a DF address expression
<l> is the length of storage to retrieve

getqwords <h>,<a>,<l>
Retrieve a one or more quadruple words from storage. If storage can’t be retrieved then a null string is returned.

<h> is a dump handle
<a> is a DF address expression
<l> is the length of storage to retrieve

format <name>,<offset>,<base>,<type>,<desc>
Format a field from a control block and returns the value of the field.

<name> is the field name.
<offset> is a relative hex offset (prefix with + or -)
<base> is the base address of the control block
<type> is the filed type (b=byte, w=word, d=double word)
<desc> is a description of the filed.

fmtblock <name>,<offset>,<base>,<type>,<number>,<desc>
Formats a table of bytes, words or double words imbedded in a control block.

<name> is the field name.
<offset> is a relative hex offset (prefix with + or -)
<base> is the base address of the control block
<type> is the filed type (b=byte, w=word, d=double word)
<number> is the number of entries in the table
<desc> is a description of the filed.

--------------------------------------------

Process Dump Formatter
PMDF provides a Process Dump Formatter facility which is invoked automatically when the Open option of the File pull-down menu is selected against a Process Dump.

The Process Dump Formatter offers a limited subset of the full Dump Formatter command set. These are:

- **.D** Display storage in Bytes, Words or Double-Words.
- **DL** Display LDT entries.
- **L** List, Symbols, Maps and Symbol Groups
- **.LM and .LMO** Display Module Table Entries and Object Tables
- **.MA** Display Arena Records for storage dumped.
- **.MO** Display Object Records for storage dumped.
- **.ML** Display Information on Dumped Memory.

**Note:**
This command does not perform the same function as the similarly named Kernel Debugger **.ML** command, which formats VM Alias Records.

- **.P** Display threads.

**Note:**
Unlike the Dump Formatter and Kernel Debugger version of this command, **.P** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.P 1** is the only valid combination.

- **.PB** Display thread Block IDs.

**Note:**
Unlike the Dump Formatter and Kernel Debugger version of this command, **.PB** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.PB 1** is the only valid combination.

- **R** Display registers for each thread.
- **.S** Set default thread slot.

**Note:**
Unlike the Dump Formatter and Kernel Debugger version of this command, **.S** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.S 1** is the only valid combination.

- **W** Load and Unload Symbol files
- **?** Syntax help for internal commands
- **.?** Syntax help for external (dot) commands.

**Note:**
Except where noted above, the command set for the Process Dump Formatter does not support any of the optional parameters supported by their equivalent Kernel Debugger commands.

When a Process Dump is loaded PMDF displays the following screen:
Note:

The data and time of the dump are displayed.

If the dump was created because of a trap then the trap number is displayed otherwise the trap number is shown as \texttt{fffffff}.

The current thread slot and register are shown last.

The Analyze pull-down menu differs from the standard PMDF Analyze facility. This offers the following choices:

**Registers**
This performs the \texttt{R} command for each thread dumped.

**Task Summary**
This performs a \texttt{.P} command followed by an \texttt{R} command for each thread dumped.

**Local Descriptors**
This performs a \texttt{DL} command.

**Virtual Memory Control Blocks**
This performs a \texttt{.MA} and \texttt{.MO} command.

**Module Table**
This is a much more extensive version of the \texttt{.LMO} command. The entire MTE and SMTE for each module dumped is formatted.

**Process Synopsis**
This formats the entire Process Dump, including dumping all memory in Byte format.

The Analyze option menu appears as follows:
For information on taking and controlling Process Dumps see:

The CONFIG.SYS DUMPPROCESS command.

The DosProcessDump API.

Kernel Debugger and Dump Formatter Command Reference

The Kernel Debugger and Dump Formatter share a common subset of commands which comprises the vast majority of the combined command set. The following symbols will be used to denote to which tool commands are applicable:

- **DF**
- **KD**

Dump Formatter

Kernel Debugger

References to some system control offsets blocks are made in the descriptions of the commands. For the sake of brevity, the ALLSTRICT version of the OS/2 WARP 3.0 kernel is assumed and in some cases the equivalent RETAIL and HSTRICT kernel offsets are given in parentheses. For example:

```
JFM_pTable (PTDA +0x5b8 (H/R: +0x5b0))
```

The reader should refer to System Reference. for control block layouts of the ALLSTRICT, HSTRICT and RETAIL kernels for OS/2 WARP 3.0 and OS/2 2.11 kernels.

Commands are categorised into two classes:

- **Internal**
  - Internal commands begin with an alphabetic character. They are control program independent in the sense that they
relate only to the Intel 80x86 hardware architecture.

External commands are prefixed with a period. They relate to the software environment under analysis and are dependent on the data structures of the operating system environment.

For a description of the conventions used in the syntax diagrams, see Syntax Diagrams - Notation

Complex expressions may be used where substituted values are required. The rules governing expressions are described in The expression evaluator.

--------------------------------------------

Syntax Diagrams - Notation

The command syntax descriptions for the Dump Formatter and Kernel Debugger use a graphical notation, which is now in common use. The diagrams should be read as a roadmap starting at the sign:

and ending at the sign:

The command verb and options are shown in upper case type.

Parameter values to be supplied by the user are shown in lower case. The rules governing the use of complex expressions are described under The expression evaluator.

Continuation of the syntax diagram is shown by:

at the break, and

at the beginning of the continuation line.

Expansion of syntax into detailed subsections is indicated thus:

section name

Expanded section begins:

section name:

Expanded section ends:

See below for a more detailed description and example of subsections.

Mutually exclusive options where a non-mandatory selection is required are shown thus:

A
B
C

Here at most one of A, B or C may be selected.
Mutually exclusive options where a mandatory choice is required are shown thus:

A
B
C

Multiple selections are shown:

D
A
E
B
F
C

One or more of A, B or C is optional, whereas at least one of D, E or F is required.

If a separator is required between parameters then it is shown in the diagram. For example, a comma is required between each selection in the following:

, ,
D
A
E
B
F
C

For the Dump Formatter and Kernel Debugger spaces between parameters options are optional.

Ordered non-exclusive selection lists and parameters are shown in the order they must be specified.

, ,
X
D = value
A
E
H
B
F
I
C

Here, X must be specified first, followed optionally by zero or more of A, B or C separated by commas, followed by at least one of D, E or F separated by commas, followed optionally by H or I and finally by the character = and a quantity substituted for value.

The following examples would be correct interpretations of this last syntax diagram:

X D = 55
XA,B I=444

Where complex diagrams require splitting into multiple sections, the sections are identified by a lower case italic name. For example:

section 1
section 2

section 1:

A
B
C

In this example the syntax for section 1 is exclusive with section 2. The options for section 1 are shown at the label section 1:
The Expression Evaluator

The Kernel Debugger and Dump Formatter expression evaluator supports a variety of arithmetic, boolean and addressing operators to form a value to be substituted into a command parameter may be derived. The atomic entities used within expressions may be string or numeric in type. Arithmetic expressions may be used with addressing separators to represent a physical, linear, selector:offset or segment:offset address's. Certain conventional values may be represented in expressions by mnemonics.

Symbols defined by symbols files may also be used to represent either their equivalent address operator and address arithmetic value combination or constant arithmetic value in command line expressions.

String Expressions

These are identified by being enclosed in either single or double quotes. A string may contain any keyboard character including quotation marks, which must be duplicated so as not to act as a string terminator. Examples are:

'\texttt{this is a sting}'
'\texttt{That''s an other example}'
"\texttt{and so is this}"

Where there is no ambiguity then terminating quote may be omitted.

Arithmetic Expressions

The expression evaluator will accept numeric values in a decimal, hexadecimal, binary and octal notation. These are indicated thus:

- \texttt{nnnnnnY}\quad \text{Binary number} \nnnnnn.
- \texttt{nnnnnnO}\quad \text{Octal number} \nnnnnn.
- \texttt{nnnnnnQ}\quad \text{Alternative notation for octal number} \nnnnnn.
- \texttt{nnnnnnT}\quad \text{Decimal number} \nnnnnn.
- \texttt{nnnnnnH}\quad \text{Hexadecimal number} \nnnnnn.

The base suffix may be in upper or lower case.
The default base when a suffix is omitted is hexadecimal.
The following represent the same number, expressed in each of the permissible forms:

\begin{verbatim}
31
31t
1f\texttt{h}
37o
37q
10001111y
\end{verbatim}

Arithmetic expressions are of three types:
Absolute
An arithmetic expression that resolves to a numeric value.

Absolute expressions may be formed from numeric values using arithmetic binary and unary operators and in-built functions together with parentheses (,), to influence evaluation order.

Boolean
Boolean expressions are ones that resolve to either a TRUE or FALSE value.

Boolean expressions may be formed from arithmetic expressions using boolean binary and unary operators together with parentheses (,), to influence evaluation order.

Boolean expressions may be used as absolute values in arithmetic expressions. Whereupon TRUE assumes the value 1 and FALSE 0.

Address
An arithmetic expression that resolves to one or two numeric values that represent a linear, physical, segment:offset or selector:offset address.

Address expressions are be formed from absolute expressions using addressing separators.

Note:
The expression evaluator allows arithmetic values to be expressed in hexadecimal. A potential conflict may occur where symbol names exist that begin with letters: a - f. For example, a linear address expressed as %fe1234 may be rejected with the message:

Invalid expression

where a symbol f or fe is defined. To avoid this conflict prefix the hexadecimal numeric value with a zero, thus:

%0fe1234

If the same error message persists then the address refers to either paged out or unallocated virtual memory.

Binary Operators

Arithmetic operators:
The following binary operators are permissible in any arithmetic expression:

*  
Multiplication

/  
Integer division

MOD  
Modulo or remainder operator

+  
Addition

-  
Subtraction

AND  
Bitwise AND

XOR  
Bitwise exclusive OR

OR  
Bitwise OR

Boolean operators:
The following binary operators are permissible in any boolean expression:

>  
Greater than

<  
Less than

>=  
Greater than or equals

==  
Logical equality

!=  
Logical inequality

&&  
Logical AND

||  
Logical OR

--------------------------------------------

Unary Operators

Arithmetic operators:

The following unary operators are permissible in any arithmetic expression:

NOT  
Bitwise ones complement

–  
Bitwise Twos complement

Boolean operators:

The following unary operator is permissible in any boolean expression:

!  
Logical negation

--------------------------------------------

In-built Functions

The following in-built functions operate in a single address expression operand:

SEG  
Returns the segment or selector portion of an address that resolves to either a &segment:offset or #selector:offset form.

OFF  
Returns the offset of an address the resolves to either a &segment:offset or #selector:offset form.

BY  
Returns one byte from an address location.

WO  
Returns one word from an address location.
Returns one double word from an address location.

**POI**

Returns one double word far pointer (selector:offset or segment:offset address) from an address location. The low order word returned is treated as the offset. The high order word returned is treated as a selector or segment based depending on the default addressing mode. See the **D command** for more information.

**PORT**

Returns one byte from an 8-bit I/O port address.

**WPROT**

Returns one word from a 16-bit I/O port address.

**Example:**

```
DD %dw(%7abcde0+10))
```

Display the storage whose linear address is at location %7abcdef0.

--------------------------------------------

**Address Separators and Address Expressions**

The following separators may be used with absolute expressions to form elements of an address:

- **&**: Segment prefix
- **#**: Selector prefix
- **%**: Linear address prefix
- **%%%**: Physical address prefix
- **:** A segment/offset address separator.
- **|**: Thread slot number qualifier.

Where virtual addresses map to different physical addresses in different processes (typically private arena data and shared arena instance data) then | may be used to qualify the address by thread slot number.

**Note:**

This qualifier is ignored by the Dump Formatter

**Examples:**

```
%ebp
```

The value of **EBP** assumed to be a linear address.

```
%%10034
```

The physical address at location 10034.

```
38  | #1f:0
```

The selector:offset address 1f:0 in the context of slot 38.
Evaluation Order

Expression are evaluated left to right by applying the following order of precedence to operators, separators and in-built functions:

1. ( )
2. | : |
3. & # % %% _ ! NOT SEG OFF BY WO DW POI PORT WPORT
4. * / MOD
5. + -
6. > < >= <=
7. == !=
8. AND XOR OR
9. && ||

Mnemonics and Symbols

Symbols defined in symbol files may be used in any arithmetic expression. Absolute symbols (that is, symbols of constants) are treated as absolute expressions. Other symbols are treated as address expressions. Symbols are activated using the WA command.

The in-built register mnemonics supported by the Kernel Debugger and Dump Formatter are:

- 16-bit registers:
  - ax, bx, cx, dx, si, di, bp, ip, pc
- 32-bit registers:
  - eax, ebx, ecx, edx, esi, edi, ebp, eip
- Segment registers:
  - cs, ds, es, fs, gs, ss
- Flag registers:
  - flg, eflg
- Control registers:
  - cr0, cr2, cr3
- GDTR register:
  - gdtb, gdtd
- IDTR register:
  - idtb, idtl
- Task control registers:
  - tr, ldtr, msw
- Debug registers:
dr0, dr1, dr2, dr3, dr4, dr5, dr6
- Test registers:
  tr6, tr7
These may be used as absolute expressions for the current register value. See the R command for information on displaying and setting current register values and for the definition of the register mnemonics.

The Kernel Debugger also defines mnemonics:
- br0, br1, br2, ..., br9
to represent the addresses of breakpoints defined by the BP and BR commands.

The expression evaluator allows the prefix @ to a symbol name to distinguish it from a similarly named mnemonic name. For example, @ax refers to the symbol ax, whereas ax refers to the ax register value.

Similar conflicts may also arise between hexadecimal values and symbols. These may be avoided by prefixing and hexadecimal numeric value with a zero.

--------------------------------------------

Internal Commands

The following comprise the set of internal commands:
- ? Display internal command help
- B Breakpoint command family.
- BC Clear breakpoint
- BD Disable breakpoint
- BE Enable breakpoint
- BL List breakpoints
- BP Set or change a breakpoint
- BR Set a debug register breakpoint
- BS Show time-stamped breakpoint trace
- BT Set time-stamped breakpoint trace
- C Compare memory
- D Dump memory data (default)
- DA Dump memory ASCII data
- DB Dump memory byte data
- DD Dump memory double-word data
- DG Dump global descriptor table
- DI Dump interrupt descriptor table
- DL Dump local descriptor table
- DP Dump Page Tables
- DT Dump Task State Segment
- DW Dump memory word data
DX  Dump 80286 Loadall buffer
E   Enter memory data
F   Fill memory
G   Go
H   Perform hexadecimal arithmetic
I   Input from 16-bit I/O port
J   Execute commands conditionally.
K   Display current stack
L   List maps, groups and symbols
M   Move memory data
O   Output to 16-bit I/O port
P   Process Trace
Q   Quit the Dump Formatter
R   Display/Alter registers
S   Search
T   Trace
U   Unassemble
V   Trap Vectors Command family
W   Add/remove symbol map
Y   Set Kernel Debugger options
Z   Set/list/execute the default command

--------------------------------------------

? - Show Internal Command Help or Evaluate an Expression

Syntax:

```
? expr string
```

Parameters:

*(default)*

Displays a help summary for most of the Dump Formatter and Kernel Debugger internal commands.

Note:

Some of the information displayed is out-of-date.
Two pages of information are displayed with an intervening **More** prompt.

### expr

An expression that resolves to either a simple numeric value or an address using any of the **expression evaluation** operators. Symbols of addresses and **symbols of absolute values** may be specified.

### string

A string enclosed in single or double quotes.

#### Results & Notes:

If an expression is specified then it is evaluated. If it resolves to an address then it is displayed in equivalent forms, as follows:

\[
\text{sel:offset \%linaddr \%physaddr}
\]

Where:

- **sel:offset**: Specifies the selector and offset form of the address if the expression resolves to a **sel:offset** form.
- **%linaddr**: Specifies the linear address equivalent of the expression if it resolves to either a **sel:offset** or **%linaddr** form.
- **%physaddr**: Specifies the physical address equivalent of the expression. If the expression resolves to a virtual address then the page tables must be present to perform the address translation.

See the **DP command** for information on displaying page table entries and the **I command** for information on paging in memory.

If the evaluated expression resolves to an absolute value then it is displayed in hexadecimal, decimal, octal, binary, character and boolean forms. For example:

```plaintext
##? 5
05H 5T 5Q 00000101Y '.' TRUE
? bmp_segsize
12H 18T 22Q 00010010Y '.' TRUE
```

#### Notes

Each arithmetic value is suffixed with a modifier that indicates the base used:

- **H**: Signifies hexadecimal
- **T**: Signifies decimal (Tens)
- **Q**: Signifies octal (Octal?)
- **Y**: Signifies binary (Yes/no?)

In the last example above, **bmp_segsize** is an **absolute symbol** of value **0x0012** defined in map **OS2KRNL**.

If a string expression is displayed then it is echoed back to the console. For example:

```plaintext
##? "This is a way of annotating the debug log from this session's analysis"
This is a way of annotating the debug log from this session's analysis
```

#### Note:

Evaluation of simple expressions involving two absolute expressions may be done using the **H command**.
B - Breakpoint Command Family

The breakpoint family of eight commands provide a means of defining and managing *sticky* breakpoints.

**Syntax:**

```
B                             C                           `          
D            options
E
L
P
R
S
T
```

**Parameters:**

**C**

Clear breakpoints.

See **BC command** for **options**.

**D**

Disable breakpoints.

See **BD command** for **options**.

**E**

Enable breakpoints.

See **BE command** for **options**.

**L**

List breakpoints.

See **BL command** for **options**.

**P**

Set or change breakpoints.

See **BP command** for **options**.

**R**

Set a debug register breakpoint.

See **BR command** for **options**.

**S**

Set a time-stamped breakpoint trace.

See **BS command** for **options**.

**T**

Display a time-stamped breakpoint trace.

See **BT command** for **options**.

**options**

See the associated command for details.
BC - Clear Breakpoints

Clear 1 or more breakpoints.

Syntax:

```
BC                            n                           *
```

Parameters:

`n`  
Breakpoint number to be cleared.

* may be specified to clear all breakpoints.

See Breakpoint commands for information on listing and setting breakpoints.

Results & Notes:
The specified breakpoints are cleared. No information is displayed.

--------------------------------------------

BD - Disable Breakpoints

Disable 1 or more breakpoints.

Syntax:

```
BD                            n                           *
```

Parameters:

`n`  
Breakpoint number to be disabled.

* may be specified to disable all breakpoints.

See Breakpoint commands for information on listing and setting breakpoints.

Results & Notes:
The specified breakpoints are disabled. No information is displayed.

--------------------------------------------

BE - Enable Breakpoints
Enable 1 or more breakpoints.

**Syntax:**

```
BE n
```

**Parameters:**

- `n` Breakpoint number to be enabled.
- `*` may be specified to enable all breakpoints.

See [Breakpoint commands](#) for information on listing and setting breakpoints.

**Results & Notes:**

The specified breakpoints are enabled. No information is displayed.

--------------------------------------------

**BL - List Breakpoints**

List all breakpoints defined by the **BP** and **BR** commands.

**Syntax:**

```
BL
```

**Parameters:** none.

**Results & Notes:**

**BL** lists the definitions of all currently defined breakpoints. An example of this follows:

```
bl
0 e 0158:00005874 [DOSOPEN] 5 (5) ".*;G"
1 d 0158:00007384 [DOSSIGNAL] 10 (10)
2 e %fff461a4 [_tkSchedNext] 12 (15) ".**
3 dT %fff474e4 [PGSwitchContext]
4 d %1a022298 [DOS32WRITE] 10 (10)
5 e W2 0030:00000000 [Ppid]
6 dI E1 0000:00000000 5 (5) "DW TASKNUMBER L1"
7 e I1 00002e7
##
```

Breakpoint definitions are of two forms:

1. The general layout of the **BP** breakpoint definition is:

   ```
   n st addr [symbol] pc (mc) "cmd, cmd, ...."
   ```

2. The general layout of the **BR** breakpoint definition is:
Each of the fields has the following meaning:

\( n \)

The breakpoint number assigned to the given breakpoint.

\( st \)

The status of the breakpoint:

- **d**: Disabled breakpoint. See BD command
- **e**: Enabled breakpoint. See BE command.

The suffix **T** signifies that the breakpoint is a time-stamp breakpoint created using the BT command.

The suffix **I** indicates that the address has become invalid.

\( tn \)

The register breakpoint type (**t**) and size (**n**):

- **R**: Read breakpoint.
- **W**: Write breakpoint.
- **I**: I/O breakpoint.

**Note**: I/O breakpoints are only available to Pentium (and later) processors. The support for I/O breakpoints was introduced into the Kernel Debugger with fix pack 29 for Warp 3.0 and base Warp 4.0.

\( addr \)

The address at which the breakpoint is defined.

\([\text{symbol}]\)

The breakpoint offset to the nearest symbolic address, if it exits. See LN and WA commands for information on listing and loading symbol definitions.

\( pc \)

The remaining **passcount** for this breakpoint. If a **passcount** is not defined then this value is not displayed. See BP and BR commands for more information on passcounts.

\((mc)\)

The initial passcount defined for this breakpoint. If no passcount was defined then this value is not displayed. See BP and BR commands for more information on passcounts.

"cmd, cmd, ...

A list of commands to be executed when the breakpoint fires. Each command is separated by commas and the entire string is enclosed in quotes. If no command string is defined then this field is not displayed. See BP, BR and Z commands for more information on breakpoint command lists.

---

**BP - Set or Alter a Breakpoint**

Set or re-specify a software **sticky breakpoint** by inserting an INT 3 instruction.

### Syntax:

```
BP addr
BPn addr passcount "cmd, cmd, ...
```


**Parameters:**

- **n**
  
  Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified. If specified there must be no space between the number and the `BP` command.
  
  The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:
  
  Too many breakpoints

- **addr**
  
  The **address** of the breakpoint. The Kernel Debugger saves the byte of storage at the location specified by `addr` and inserts an INT 3 instruction in its place.

**Notes**

Whenever the Kernel Debugger is entered the storage overlayed by any breakpoints is temporarily restored. When the Kernel Debugger gives control back to the system, enabled breakpoints are re-instated.

If `addr` specifies the address of an existing breakpoint then the existing breakpoint is updated with the new parameters.

Each break-point address is recorded with its associated process context. For shared data this is of no consequence. However for private addresses, especially those in the private arena the `addr` may be qualified by `slot` number by using the `|` operator. This acts as a shorthand to save changing contexts using the `.S command` in order to set the breakpoint correctly. For example, suppose the current slot is 8, then:

```
BP 31 | %10032
```

is equivalent to:

```
.S 31
BP 31 | %10032
.S 8
```

- **passcount**
  
  Specifies the number of times the breakpoint may be passed before the Kernel Debugger is entered. Each time the breakpoint is passed the count is decremented by 1 until 1 is reached. When the breakpoint is encountered with a count of 1 then it will fire and the Kernel Debugger will be entered. Thus if `passcount` is 5 then the breakpoint will fire on the 5th encounter.

  The default `passcount` is 1, that is, the breakpoint will fire on first encounter.

- **cmd**
  
  Specifies a command to be executed when the breakpoint fires. More than one command may be specified by using a semi-colon separator and enclosing the entire command list in single or double quotes.

  If no command string is specified then the default command string, as specified by the `Z command` will be executed.

**Results & Notes:**

- If the specified address is valid then the breakpoint definition is accepted otherwise one of the following messages is generated:
  
  - **Invalid linear address: %nnnnnnnn**
  - **Invalid selector: selector:offset**
  - **Past end of segment selector:offset**

- If the break-point is successfully defined then the in-built mnemonic `BRn`, where `n` corresponds to the break-point number, takes the value of the break-point address. This may be used in any address expression or any command.

**Note:**

Since `BP` break-points are implemented by the insertion of INT 3 instructions, it is possible for such break-points to become discarded if the page of code is discarded and subsequently paged back into memory.
If the `.I` command is used to swap in a page of code, then the break-points are automatically restored. (In earlier versions of OS/2 it was necessary to specify the `B` option of `.I`).

This complexity may be avoided by setting register break-points with the `BR` command.

BR - Set or Alter a Debug Register Breakpoint

Set or alter a _sticky_ break-point, using the debug registers.

**Syntax:**

```
BR        E      addr                                    `  
BRn       Wb              pc         ,     cmd
Rb                       , "     cmd     
Ib
```

**Parameters:**

`n`  
Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified but from this range only a total of 4 may specify enabled debug register breakpoints.

If a value `n` is specified there must be no space between the number and the `BR` command.

The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:

Too many breakpoints

If all four debug registers are in use then the message:

Out of debug registers

is displayed.

**Note:**

A disabled debug register breakpoint does not commit the use of a debug register. Thus more that 4 debug register breakpoints may be defined, but only a maximum of 4 enabled at any time.

See the `BE` and `BD` commands for information on enabling and disabling breakpoints.

`E`  
Specifies that the breakpoint is to _fire_ when an instruction at the breakpoint address is fetched for execution.

This is mutually exclusive with the `W` and `R` parameters.

`Rb`  
Specifies that the breakpoint is to _fire_ when storage at the breakpoint address, for length `b` is referenced. `b` may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

This is mutually exclusive with the `W`, `I` and `E` parameters.

`Wb`  
Specifies that the breakpoint is to _fire_ when storage at the breakpoint address, for length `b` is stored. `b` may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

This is mutually exclusive with the `R`, `I` and `E` parameters.
Specifies that the breakpoint is to fire when data is read or written from an I/O port address specified by `<addr>`. The length operand, `<b>` is used by the processor to mask out the low order bits of the I/O address. Thus:

```
BR I2 %13f
```

places an I/O breakpoint on ports 13c - 13f. `<b>` may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

Note that the I/O port address must be specified as a linear address, otherwise the expression evaluator will attempt to use the base address of the current CS register to resolve the address parameter.

This is mutually exclusive with the R, W and E parameters.

**Note:** I/O breakpoints are only available to Pentium processors. Support for this was introduced from fix pack 29 of Warp 3.0 and base Warp 4.0, however a bug has prevented them from working correctly until Fix Pack 0 of Warp 4.0 and Fix Pack 39 of Warp 3.0.

**addr**

The **address** of the breakpoint.

The Kernel Debugger converts the address to a linear address before setting up the debug registers. If the address is invalid the definition is retained but marked disabled and invalid.

**Note:** Real addresses may not be used with debug register breakpoints.

**passcount**

Specifies the number of times the breakpoint may be passed before the Kernel Debugger is entered. Each time the breakpoint is passed the count is decremented by 1 until 1 is reached. When the breakpoint is encountered with a count of 1 then it will fire and the Kernel Debugger will be entered. Thus if `passcount` is 5 then the breakpoint will fire on the 5th encounter.

The default passcount is 1, that is the breakpoint will fire on first encounter.

**cmd**

Specifies a command to be executed when the breakpoint fires. More than one command may be specified by using a semi-colon separator and enclosing the entire command list in single or double quotes.

If no command string is specified then the default command string, as specified by the Z command will be executed.

**Note:** The command list must be preceded by a comma, unlike the BP command where the comma is optional.

**Results & Notes:**

If the specified address is valid then the breakpoint definition is accepted and enabled otherwise it is accepted but disabled and one of the following messages is generated:

```
Invalid selector: selector:offset
Past end of segment selector:offset
```

If the breakpoint is successfully defined then the in-built mnemonic BRn, where n corresponds to the break-point number, takes the value of the break-point address. This may be used in any address expression or any command.

BS - Show Time-stamped Breakpoint Trace

Show the time-stamped breakpoint trace.

**Syntax:**

```
BS
```
**Parameters:**

None.

**Results & Notes:**

The time-stamp trace buffer is formatted in LIFO order. The following is an example of the formatted trace:

Number of entries = 4284
BP0 381e6a1a (hex)
BP4 381e68292 (hex)
BP0 381e658d1 (hex)
BP4 381e40559 (hex)
BP0 381e3da7d (hex)

**Notes**

The number of entries is the total accumulated number of time-stamp trace events regardless of wrapping of the (4096 entry) time-stamp trace buffer.

Each entry show the breakpoint number that corresponds to the time-stamped breakpoint that fired, the high resolution time stamp in microseconds and a reminder that this value is in hexadecimal.

For information on defining a time-stamp breakpoint see the BT command.

--------------------------------------------

**BT - Set Time-stamped Breakpoint Trace**

Set a time-stamped breakpoint trace.

**Syntax:**

```
BT addr
BTn
```

**Parameters:**

- **n**: Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified. If specified, there must be no space between the number and the BT command.

  The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:

  Too many breakpoints

- **addr**: The address of the breakpoint.

  The Kernel Debugger saves the byte of storage at the location specified by addr and inserts an INT 3 instruction in its place.

**Notes**

Whenever the Kernel Debugger is entered the storage overlayed by any breakpoints is temporarily restored. When the Kernel Debugger gives control back to the system, enabled breakpoints are re-instated.

If addr specifies the address of an existing breakpoint then the existing breakpoint is updated with the new parameters.
Results & Notes:

If the address is valid then the breakpoint definition is accepted and enabled. When enabled, the time-stamp breakpoint causes the current high resolution system time to be saved in a time-stamp circular trace buffer whenever the breakpoint address is executed.

The trace buffer will record up to 4K entries before wrapping.

Unlike the BP and BR commands, BT does not return control to user when the breakpoint is encountered.

The time-stamp trace may be displayed using the BS command.

C - Compare Memory

Compare up to 64k bytes of memory at two locations in storage.

Syntax:

C       addr1        n       addr2

Parameters:

addr1
The address of the beginning of the first location to compare with the second. This address is assumed to be in \#selector:offset format. If the selector is omitted then the current DS selector is assumed.

n
The offset from addr1 of the last byte to compare (that is, the length of the range less 1).

addr2
The address of the beginning of the second location to compare with the first. An address expression may be specified. This address is assumed to be in \#selector:offset format. If the selector is omitted then the current DS selector is assumed.

Results & Notes:

Storage is compared, if no differences are found then the command prompt is displayed. If either of the addresses is invalid then an error message is displayed.

Where differences are found in the address range then they are displayed in the following way:

001f:00000000 57 4f 001f:00000003
001f:00000001 50 32 001f:00000004
001f:00000003 57 4f 001f:00000006

The addresses of the two differing locations are displayed outermost and the bytes at those locations are displayed in columns 2 and 3.

D - Display Memory

Display a range of memory from a given address.

Syntax:

D

DA           addr
DB                     Ln
Parameters:
(default)
Display memory using the current display format. When the user breaks into the Kernel Debugger the current format is set to Byte display. If the user subsequently executes a **DW**, **DA** or **DD** command then the current format is set to words, ASCII or double-words, respectively. Byte format default may be restored by using **DB**.

A

Force memory to be displayed in ASCII format and set the current display format to ASCII. The display is terminated as soon as the first null byte (0x00) is reached or the length specification is reached.

Note: The current display address is not updated when in ASCII format.

B

Force memory to be displayed in Byte format and set the current display format to Byte.

W

Force memory to be displayed in Word format and set the current display format to Word.

D

Force memory to be displayed in Double-word format and set the current display format to Double-word.

`addr`

The address of the memory location to display. When the user breaks into the Kernel Debugger this defaults the current **DS** selector, offset 0. If a display command other than **DA** is executed then the current display address is updated to the last displayed address +1.

An address expression may be specified.

`Ln`

The number of bytes, words or double-words to display, depending upon the current display format. If not specified this defaults to 128 bytes, 64 words and 32 double-words respectively.

Results & Notes:

Memory is displayed according to the selected display format providing the address is valid. If it is not, but the address represents pageable storage then this may be paged in to memory using the `.I` command.

The following examples show output in the four different formats.

ASCII format:

```
##da 1f:0
001f:00000000 WP_OBJHANDLE=177110
```

Byte format:

```
##db 1f:0
001f:00000000 57 50 5f 4f 42 4a 48 41-4e 44 4l 3d 31 37 37 WP_OBJHANDLE=177
001f:00000010 31 31 30 00 55 53 45 52-5f 49 4e 49 3d 43 3a 5c 110.USER_INI=C:\
001f:00000020 4f 53 32 5c 4f 53 32 2e-49 4e 49 00 53 59 53 54 OS2\OS2.INI
001f:00000030 4c 4f 53 32 5c 4f 53 32-4e 45 5f 49 4e 49 3d 43 3a 5c 49 4f 53_2\OS2\CMD.EX
001f:00000040 45 00 41 55 54 4f 53-41 52 54 3d 54 41 53 4b E.AUTOSTART=TASK
001f:00000050 4c 49 53 54 2c 46 4f-4l 44 45 58 00 52 45 53 LIST,FOLDERS.RES
```
DA - Display Memory in ASCII Format

Display a range of memory from a given address in ASCII format.

Syntax:

```
DA addr Ln
```

See the D command for a full description.

DB - Display Memory in Byte Format

Display a range of memory from a given address in Byte format.

Syntax:

```
DB addr Ln
```
See the D command for a full description.

--------------------------------------------

**DW - Display Memory in Word Format**

Display a range of memory from a given address in Word format.

**Syntax:**

```
DW addr Ln
```

See the D command for a full description.

--------------------------------------------

**DD - Display Memory in Double-word Format**

Display a range of memory from a given address in Double-word format.

**Syntax:**

```
DD addr Ln
```

See the D command for a full description.

--------------------------------------------

**DG - Display Global Descriptor Table**

Display entries from the Global Descriptor Table.

**Syntax:**

```
DG DGA s Ln
```

**Parameters:**

(Default) Display valid GDT entries only.
Display all GDT entries including invalid descriptors.

Display descriptor for selector number \( s \).

Notes

Since bit 2 of the selector determines whether the descriptor is local or global the correct table entry will be displayed regardless of whether the \( DG \) or \( DL \) command is used. If an LDT descriptor is specified then the following message is displayed:

\[ \text{LDT} \]

The requestor priority level (RPL) bits (bits 0 and 1 of the selector) are ignored by \( DG \). Thus: \( DG \ 8 \) displays the same information as \( DG \ 9, DG \ a \) and \( DG \ b \).

If the \( s \) parameter is omitted then the entire GDT is displayed.

The number of descriptor entries to display from and including selector \( s \). The default is to display one descriptor entry.

Results & Notes:

One or more descriptor table entries are displayed. An example display follows:

```plaintext
##dga
0000  Invalid Bas=00000000 Lim=00000000 DPL=0 NP
0008  Invalid Bas=00000000 Lim=00000000 DPL=0 NP
0010  TSS32 Bas=ffe05dfc Lim=00000067 DPL=0 P B
0018  TSS32 Bas=ffe05dfc Lim=00000067 DPL=0 P B
0020  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0028  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0030  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0038  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0040  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0048  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0050  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0058  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0060  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
0068  Data Bas=ffe05dfc Lim=00000067 DPL=0 P B
```

For a detailed explanation of the descriptor table entry format see Description Table Entry Format

```
Descriptor Formats

The Kernel Debugger and Dump Formatter format descriptor table entries in either of two forms depending on whether the descriptor describes a segment of memory or a gate:

\[ \text{dddd \ type Bas=bbbbbbbb Lim=llllllll DPL=p flags} \]

\[ \text{dddd \ type Sel:Off=ssss:ooooooo DPL=p flags} \]

Each of these fields has the following meaning:

\[ \text{dddd} \]

Descriptor number

\[ \text{type} \]

Descriptor type. The following are defined:
<table>
<thead>
<tr>
<th>Type</th>
<th>Type Numbers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>-</td>
<td>Code segment</td>
</tr>
<tr>
<td>Data</td>
<td>-</td>
<td>Data segment</td>
</tr>
<tr>
<td>Invalid</td>
<td>0 or 8</td>
<td>Invalid descriptor</td>
</tr>
<tr>
<td>TSS</td>
<td>1 or 3</td>
<td>Available or Busy 80286 TSS</td>
</tr>
<tr>
<td>LDT</td>
<td>2</td>
<td>System descriptor for an LDT</td>
</tr>
<tr>
<td>CallG</td>
<td>4</td>
<td>Call Gate</td>
</tr>
<tr>
<td>TaskG</td>
<td>5</td>
<td>Task Gate</td>
</tr>
<tr>
<td>IntG</td>
<td>6</td>
<td>80286 Interrupt Gate</td>
</tr>
<tr>
<td>TrapG</td>
<td>7</td>
<td>80286 Trap Gate</td>
</tr>
<tr>
<td>Reserve</td>
<td>10 or 13</td>
<td>Reserved descriptor types</td>
</tr>
<tr>
<td>TSS32</td>
<td>9 or 11</td>
<td>Available or Busy Intel486 CPU TSS</td>
</tr>
<tr>
<td>CallG32</td>
<td>12</td>
<td>Inter486 CPU Call Gate</td>
</tr>
<tr>
<td>IntG32</td>
<td>14</td>
<td>Intel486 CPU Interrupt Gate</td>
</tr>
<tr>
<td>TrapG32</td>
<td>15</td>
<td>Intel486 CPU Trap Gate</td>
</tr>
</tbody>
</table>

**Bas=bbbbbbbb**

Segment base address.

**Lim=llllllll**

Segment limit address.

**DPL=p**

Descriptor priority level. Only 0, 2 and 3 are used in OS/2.

**Sel=ssss:Off=00000000**

*selector:offset* transfer address for a task, interrupt, trap or call gate descriptor.

**flags**

Interpretation of the various descriptor flags. The following abbreviations are used:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>~15</td>
<td>Not present</td>
</tr>
<tr>
<td>P</td>
<td>15</td>
<td>Present</td>
</tr>
<tr>
<td>RW</td>
<td>9</td>
<td>Read/Write data segment</td>
</tr>
<tr>
<td>RO</td>
<td>~9</td>
<td>Read-only data segment</td>
</tr>
<tr>
<td>ED</td>
<td>10</td>
<td>Expand-down data segment</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>Conforming code segment</td>
</tr>
<tr>
<td>G4k</td>
<td>23</td>
<td>4K granularity segment limit</td>
</tr>
<tr>
<td>BIG</td>
<td>22</td>
<td>32-bit Stack offsets (ESP) used as a stack segment. (No meaning when used as a data segment).</td>
</tr>
<tr>
<td>C32</td>
<td>22</td>
<td>32-bit operands and data</td>
</tr>
</tbody>
</table>
sizes used by default with this code segment

RES  21 reserved
UV   20 Available bit. Used in OS/2 to indicate a UVIRT mapping.
WC=w  0 Word count of a 16-bit call gate
DWC=w 0 Double-word count of a 32-bit call gate
RE  9 Read/Execute code segment
EO ¬9 Read-only code segment
A   8 Code or Data segment accessed
NB  ¬ TSS/TSS32 not busy (available)
B   ¬ TSS/TSS32 busy

Notes

The bit offsets given above are relative to the second double-word of the descriptor viewed as 2 double-words. The INTEL programmer's Reference shows the descriptor format as a quad-word, but uses the same offsets specified above.

See the INTEL Pentium User's Reference or the INTEL x86 Programmer's References for further information.

DI - Display Interrupt Descriptor Table

Display entries from the Interrupt Descriptor Table.

Syntax:

```
DI
DIA          i
Dia          Ln
```

Parameters:

(Default)

Display valid IDT entries only.

A

Display all IDT entries including invalid descriptors.

i

Display descriptor for interrupt vector i.

Ln

The number of descriptor entries to display from and including selector i. The default is to display one descriptor entry.
Results & Notes:

One or more descriptor table entries are displayed. An example display follows:

```
##dia
0000  TrapG32 Sel:Off=0170:fff47e64     DPL=0 P
0001  IntG32  Sel:Off=0170:fff47f10     DPL=3 P
0002  TaskG  Sel:Off=1e38:00000000     DPL=0 P
0003  IntG32  Sel:Off=0170:fff480cc     DPL=3 P
0004  TrapG32 Sel:Off=0170:fff48158     DPL=3 P
0005  TrapG32 Sel:Off=0170:fff48164     DPL=0 P
0006  TrapG32 Sel:Off=0170:fff48170     DPL=0 P
0007  TrapG32 Sel:Off=005a:1a090911     DPL=0 P
0008  TaskG  Sel:Off=0088:00000000     DPL=0 P
0009  TrapG32 Sel:Off=0170:fff48258     DPL=0 P
000a  TrapG32 Sel:Off=0170:fff48268     DPL=0 P
000b  TrapG32 Sel:Off=0170:fff48270     DPL=0 P
000c  TrapG32 Sel:Off=0170:fff48278     DPL=0 P
000d  TrapG32 Sel:Off=0170:fff48280     DPL=0 P
000e  TrapG32 Sel:Off=0170:fff4853c     DPL=0 P
000f  TrapG32 Sel:Off=0170:fff48544     DPL=0 P
0010  TrapG32 Sel:Off=0170:fff4854c     DPL=0 P
```

For a detailed explanation of the descriptor table entry format see [Descriptor Table Entry Format](#).

----------------------

**DL - Display the Current Local Descriptor Table**

Display entries from the Local Descriptor Table of the default thread slot. See the `.S command` for information of changing the default thread slot.

**Syntax:**
```
DL
DLA s
DLP Ln
DLS
DLH
```

**Parameters:**

*(Default)*

- **Display valid LDT entries only.**

- **A**
  - Display all LDT entries including invalid descriptors.

- **P**
  - Obsolete option. Was used to display only valid private arena LDT descriptors where bits 3 and 4 of the selector number are 0.

- **S**
  - Obsolete option. Was used to display only valid shared arena LDT descriptors where bits 3 and 4 of the selector number are non-zero.

- **H**
  - Obsolete option. Was used to display only huge segment LDT descriptors.

- **s**
  - Display descriptor for selector number `s`.

**Notes**
Since bit 2 of the selector determines whether the descriptor is local or global the correct table entry will be displayed regardless of whether the DL or DG command is used. If an GDT descriptor is specified then the following message is displayed:

GDT

The requestor priority level bits (bits 0 and 1 of the selector) are ignored by DL. Thus DL 7 displays the same information as DL 6, DL 5 and DL 4.

If the $ parameter is omitted then the entire LDT is displayed.

Results & Notes:

One or more descriptor table entries are displayed. An example display follows:

```
##dl
0007 Data Bas=7ab27000 Lim=0000ffff DPL=3 P RO
000f Code Bas=00010000 Lim=000005ff DPL=3 P RE
0017 Data Bas=00020000 Lim=0000005b DPL=3 P RW
001f Data Bas=00030000 Lim=0000fa1f DPL=3 P RW A
0027 Data Bas=00040000 Lim=00000276 DPL=3 P RW A
002f Data Bas=00050000 Lim=00000fff DPL=3 P RW
0036 Data Bas=00060000 Lim=00003fff DPL=2 P RW A
003f Data Bas=00070000 Lim=00000fff DPL=3 P RW A
0047 Data Bas=00080000 Lim=00000fff DPL=3 P RW
004f Data Bas=00090000 Lim=00000fff DPL=3 P RW A
0056 Code Bas=000a0000 Lim=00000af7 DPL=2 P RE C
005f Data Bas=000b0000 Lim=00000fff DPL=3 P RW
```

For a detailed explanation of the descriptor table entry format see Descriptor Table Entry Format

DP - Display Page Directory and Table Entries

Display entries from the page tables of the default thread slot. See the .S command for information on changing the default thread slot.

Syntax:

```
DP
DPD addr
DPA Ln
```

Parameters:

A

Display both page table and page directory entries. This is the default.

D

Display only page directory entries.

addr

The linear or virtual address whose page directory and table entries are to be displayed. If not specified DP displays the entire page directory and its page tables.

An address expression may be specified.
The number of page table entries to display starting with the entry for `addr`. The default is to display all page table entries from this entry assigned to `addr`.

**Note:** Due to a bug in some versions of the Kernel Debugger an extra zero is required for this parameter.

**Results & Notes:**

One or more page and directory table entries are displayed. An example display follows:

```plaintext
DP %90000 L50
linaddr  frame  pteframe  state  res  Dc  Cd  WT  Us  rW  Pn  state
%00090000* 012f3  frame=012f3  2  0  D  A  U  W  P  resident
%00090000  vp id=00a76  0  0  c  u  U  W  n  pageable
%000a0000  00b8  vp id=00b88  1  0  D  u  U  W  n  uvirt
%000b0000  0088  frame=00888  0  0  D  A  U  W  P  pageable
%000c0000  vp id=00b8f  0  0  c  u  U  W  n  pageable
%000d0000  vp id=00b92  0  0  c  u  U  W  n  pageable
##
```

Output from the DP command is presented in tabular form. Each of the columns shown is described as follows:

**linaddr**

Linear address of virtual memory whose page directory and table entries are being formatted. Those lines corresponding to directory entries have an * flag suffixed to the linear address. Page table entries for a given directory entry are formatted following the directory entry.

In the example above linear address `%90000` has its page table located in physical frame `12f3`, that is at physical `%%12f3000`. The page table entry corresponding to virtual memory at `%90000` is described in the second line. Each of the following lines are consecutive entries from page table `12f3`.

**frame**

The real storage frame number that contains either the page table (* suffix to linaddr) or page frame corresponding to the linaddr. If this field is blank then the frame has been discarded. If it contains a frame number then the contents are still valid even though the page table entry no longer points to a page frame. See pteframe field for further discussion.

**pteframe**

For table entries with the present bit set the this field shows the page frame number pointed to by this table entry. This is shown as `frame=fffff`. Use the frame number with the .MP command to obtain information on allocation and ownership of this frame of real storage.

For decommitted pages the table entry contains the Virtual Page ID. This is shown as `vp id=vvvvv`. Use the .MV command with the virtual page Id to obtain information on allocation and ownership of this memory.

**state**

State information is stored in the available bits (9 - 11) of the page table entry. These are interpreted on the right-hand end of the display. The following values may appear:

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageable</td>
<td>0</td>
<td>Storage may be paged-out to the swap file</td>
</tr>
<tr>
<td>uvirt</td>
<td>1</td>
<td>Physical to virtual mapping reservation only.</td>
</tr>
<tr>
<td>resident</td>
<td>2</td>
<td>Non-pageable fixed storage</td>
</tr>
<tr>
<td>uvirt</td>
<td>3</td>
<td>Physical to virtual</td>
</tr>
</tbody>
</table>

**Notes**

The `vp id` is not valid to use with .MV if the state of the table entry is uvirt.

If the frame has been decommitted but the frame field still shows a frame number then the frame contents are still valid for reclaiming without a page-in operation from the swap file. The corresponding virtual page will be queued from the idle list. See .MV and .MP commands for more information on page management.
Reserved page table entry bits. Should always be zero.

Set to D is the page is dirty, otherwise c (clean).

Set to A if the page has been accessed, otherwise u (unaccessed).

Set to CD is the TLB cache-disable bit is set, else blank.

Set to WT is the TLB cache write-transparent bit is set, else blank.

Set to U if the page is for user storage, otherwise s (supervisor).

Set to r is the page is read-only, otherwise W (writeable).

Set to P if the page is present, otherwise n (not present).

The Dump Formatter does not format page directory entries correctly. For page directory entries only the frame field is correct. The remaining fields are taken from the first PTE of the page table associated with the page directory. This problem has been fixed from fix pack 36 for Warp 3.0 and fix pack 7 for Warp 4.0.

Refer to the following for more information on page and memory management: .M family of Kernel Debugger and Dump Formatter commands. Intel Pentium User's Guide and Intel x86 Programmer's Reference.

DT - Display a Task State Segment

Format a task state segment.

Syntax:

```
DT addr
```

Parameters:

`addr`

The address of the task state segment to be formatted. If not specified then the current TSS pointed to by the TR (task register) is used.

Results & Notes:

The TSS is formatted as follows:
Each of the fields displayed has the following meaning:

Note: Some of the TSS fields are set at task creation and other when a task switch occurs.

eax= Saved EAX register when a task switch occurs.

ebx= Saved EBX register when a task switch occurs.

ecx= Saved ECX register when a task switch occurs.

dx= Saved EDX register when a task switch occurs.

esi= Saved ESI register when a task switch occurs.

di= SavedEDI register when a task switch occurs.

eip= Saved EIP register when a task switch occurs.

esp= Saved ESP register when a task switch occurs.

ebp= Saved EBP register when a task switch occurs.

iopl= Saved EFLAGS iopl and flag settings when a task switch occurs. See .R command for an explanation of the flag abbreviations.

cs= Saved CS register when a task switch occurs.

ss= Saved SS register when a task switch occurs.

ds= Saved DS register when a task switch occurs.

es= Saved ES register when a task switch occurs.

fs= Saved FS register when a task switch occurs.

gs= Saved GS register when a task switch occurs.

cr3= CR3 register at task creation.

Note: This provides the real address of the Page Directory Table, which never alters under OS/2.
Ring 0 SS register used for ring 0 privilege transitions.

\[ \text{esp0} = \] Ring 0 ESP register used for ring 0 privilege transitions.

\[ \text{ss1} = \] Ring 1 SS register used for ring 1 privilege transitions.

\[ \text{esp1} = \] Ring 1 ESP register used for ring 1 privilege transitions.

**Note:** Ring 1 is not used under OS/2.

\[ \text{ss2} = \] Ring 2 SS register used for ring 2 privilege transitions.

\[ \text{esp2} = \] Ring 1 ESP register used for ring 2 privilege transitions.

\[ \text{ldtr} = \] LDTR register at task creation.

\[ \text{link} = \] TR register value of previous nested task's TSS.

\[ \text{tflags} = \] The debug trap bit for this task.

\[ \text{i/o maps} = \] Offset to the I/O permission map from the beginning of this TSS.

**Note:** It is permissible for the i/o map offset to point beyond the TSS segment. This signifies that no I/O permissions are granted and all ports will be trapped.

\[ \text{ports trapped:} \] List the range of I/O port addresses that will generate traps if accessed by this task.

**Notes:**

For performance reasons hardware implemented task switching is used only in a limited way in OS/2. TSSs defined by OS/2 include:

- Protect mode code (TSS selector 10)
- Virtual DOS Machines
- Non Maskable Interrupt handling (trap 2, TSS selector 1E38)
- Double Fault handling (trap 8, TSS selector 88)

All protect-mode processes run under a common top-level task using selector 10 as the TSS selector.

The `selts` field of the PTDA records the top-level task's TSS selector used by a given process, thus may be used to find the TSS selector for Virtual DOS Machines.

Refer to Intel Pentium User's Guide and Intel x86 Programmer's Reference for more information on the Task State Segment and Hardware architected multi-tasking.

--------------------------------------------

**DX - Display the 286 LoadAll Buffer**

\[ \text{DX} \]

Formats a the 286 LoadAll buffer from physical address %%800 in memory.

**Syntax:**
DX

Parameters:
None.

Results & Notes
This command applies to the Intel 286 processor and is now obsolete. The results are meaningless.

--------------------------------------------

E - Enter Data Into Memory

Enter data into a memory location.

Syntax:

```
E addr value
```

Parameters:

- **addr**: The address of the memory location to be changed. If not specified this defaults to DS:00000000 where DS is established by the most recent register display. See An address expression may be specified. R and .R commands for information on establishing default addresses.

- **value**: A numerical byte value to be entered into memory. One or more values may be specified separated commas or blanks. These may be mixed with "string" values.

- **string**: A character sting enclosed in quotes. Each character is treated as a byte value and entered into memory separately, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical values.

Results & Notes:

If memory is present values are entered into storage otherwise an Invalid Address message is generated. If this should happen, valid storage may be paged into memory by means of the .I command.

If no value or string parameter is specified the Kernel Debugger prompts the user a byte at a time for replacement values by displaying the original value followed by a colon. In prompt mode, the user may proceed as follows:

- type a replacement byte value in hexadecimal, or
- accept the original value and move on to the next location by pressing the space-bar, or
- back up to the previous location by entering a - (minus) character, or
- terminate prompt mode by pressing carriage return (with or without a replacement value).

--------------------------------------------

F - Fill Memory With Repeated Data
Fill memory with repeated data.

Syntax:

```
F          addr       Ln               value              `,
```

**Parameters:**

- **addr**
  - The address of the memory location to be changed.
  - An address expression may be specified.

- **Ln**
  - The number \( n \) of bytes to fill with data.

- **value**
  - A numerical byte value to be entered into memory. One or more values may be specified separated commas or blanks. These may be mixed with "string" values.

- **string**
  - A character sting enclosed in quotes. Each character is treated as a byte value and entered into memory separately, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical values.

**Results & Notes:**

The list of values and strings is repeated up to the length \( L_n \) and used to fill memory at the specified address. If the fill data is shorter than the length then it is repeated, if it is longer it is truncated.

If memory is present the storage is updated otherwise an Invalid Address message is generated. If this should happen, valid storage may be paged into memory by means of the .I command.

```
G - Go
```

Cause execution to continue from a given point and optionally set 1 or more 'go' breakpoints.

**Syntax:**

```
G
GS          =     start-addr           break-addr
GT
```

**Parameters:**

- (Default)
  - Continue execution from the current CS:EIP.
The **go-special** command causes the high-resolution time interval to be recorded from the point **GS** command is issued to the point that the Kernel Debugger is re-entered as the result of a break-point firing.

**Notes**

- No account is taken of the Kernel Debugger overhead when calculating the time interval.
- When the Kernel Debugger re-enters, for whatever reason, the interval timer is cancelled until another **GS** command is executed.
- If the reason for entry is for reasons other than the firing of a sticky or go breakpoint then in addition to cancelling the interval timer no time message displayed.

**T**

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the **IDT** and the system's re-instated until after then next instruction has executed. After execution of the next instruction the the Kernel Debugger's **V commands** are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the **IDT** using a command sequence similar to:

```
VC n
T
VS n
G
```

**start-addr**

The address from which execution is to continue. This must be a valid address for the current context. If **start-addr** is omitted then execution continues from the current **CS:EIP**, as shown by the **R command**.

**Warning:** Be very careful to ensure that the start address is valid for the privileged level and addressability of the code and data selectors in use. If the Kernel Debugger attempts to load a segment register that is invalid the system may trap in the debugger code.

**break-addr**

Up to ten **go breakpoints** may be specified. These are temporary breakpoints set in addition to any sticky breakpoints set by the **B commands**. When the Kernel Debugger is next entered, for whatever reason, all **go** breakpoints are cleared.

If **break-addr** is omitted then the system continues execution until:

- A fatal exception occurs
- An Internal Processing Error (IPE) occurs.
- A sticky breakpoint fires
- A non-maskable interrupt occurs
- An **INT 3** instruction is executed
- The user enters **Ctrl-C** from the debugging console.

**Results & Notes:**

The system continues execution until the Kernel Debugger is re-entered. If the reason for entry is other than a breakpoint firing then the **R command** is automatically executed followed by one of the following command prompts:

- **>** (signifies a command prompt in real mode)
- **#** (signifies a command prompt in protect mode with paging disabled)
- **-** (signifies a command prompt in V86 mode with paging disabled)
- **##** (signifies a command prompt in protect mode with paging enabled)
- **--** (signifies a command prompt in V86 mode with paging enabled)

If an error situation caused entry to the Kernel Debugger then a diagnostic message may be generated by the failing code writing directly to the Kernel Debugger's communications port.
If entry was caused by a Kernel Debugger trap handler receiving control then a message from the trap handler will be displayed. See the V command for details.

If a breakpoint caused the Kernel Debugger to receive control then commands associated with the breakpoint that fired will execute. See the B commands for details.

If a go-special was interrupted by a breakpoint firing then the following message appears before any output associated with the breakpoint:

Go Time (tics) = 017fb (hex) = 5145 (uSec)

This shows the time interval in both timer-ticks and equivalent number of micro-seconds.

H - Hex Arithmetic

Display the sum, difference, product, quotient and remainder of two absolute expressions.

Syntax:

```
H        abs-expr1       abs-expr2
```

Parameters:

- **abs-expr1**
  - An expression that resolves to a simple numeric value using any of the expression evaluator operators. Symbols of absolute values may be specified in the expression, but symbols of relocatable addresses may not.

- **abs-expr2**
  - An expression that resolves to a simple numeric value using any of the expression evaluator operators. Symbols of absolute values may be specified in the expression, but symbols of relocatable addresses may not.

Results & Notes:

Each of the expressions is evaluated. If either does not resolve to a simple numeric value then the message:

Expression error

is displayed.

Having resolve each of the expressions then the sum, difference, product and quotient of the pair is displayed as in the following examples:

```
##h 2 3
+0005  -ffff  *0006 0000 /0000 0002
##h 10c 5
+000f  -0005  *0032 0000 /0002 0000
##h 7fff 5
+8004  -7fda  *7ff8 0002 /1999 0002
##h 5*4 2*3
+001a  -000e  *0078 0000 /0003 0002
##h bmp_segsize 5
+0017  -000d  *005a 0000 /0003 0003
```

Notes

Calculations are performed using 16-bit signed arithmetic.
The operation performed is shown prefixing the result.
The product is shown as a two word value, the low word followed by the high.
The division is shown as two words, the quotient followed by the remainder.
In the last example, bmp_segsiz is an absolute symbol of value 0x0012 defined in map OS2KRNL.
Evaluation of complex expressions involving relocatable addresses may be done using the ? command.

--------------------------------------------
I - Input From an I/O Port

Input a byte of data from a 16-bit I/O Port

Syntax:

I port `.

Parameters:

port
A 16-bit I/O port address. This may be specified as a simple numeric expression.

Results & Notes:
The byte of data is read from the requested I/O port and displayed in hexadecimal at the console. For example:

#I 2f8
0d

See also the O command for related information.

--------------------------------------------
J - Execute Commands Conditionally

Conditionally execute one of two lists of commands depending on whether an expression evaluates to TRUE (non-zero) or FALSE (zero).

Syntax:

J expression ` ; cmd1 ; cmd2

Parameters:
**expression**

An expression that resolves to either a simple numeric value or an address using any of the expression evaluation operators. Symbols of addresses and symbols of absolute values may be specified.

**cmd1**

Specifies a command to be executed if the expression evaluates to **TRUE** (non-zero). More than one command may be specified if each is separated by a semi-colon and the entire command list is enclosed in single or double quotes.

If **cmd1** is omitted, control is returned to the debugging console when the expression is **TRUE**.

**cmd2**

Specifies a command to be executed if the expression evaluates to **FALSE** (non-zero). More than one command may be specified. Each cmd2 must be prefixed by a semi-colon, even if only one is specified. Quotes are not required to encompass a list of

If **cmd2** is omitted, control is returned to the debugging console when the expression is **FALSE**.

**Results & Notes:**

If the expression resolves to one of the following forms, it is considered to be **FALSE**:

- 0
- 0:0
- &0:0
- %0
- %%0

Any other resolution is regarded as **TRUE**.

The **J** command is primarily intended to be used with the **BP** and **BR** commands to enable conditional **breakpoints** to be defined.

Examples of this usage are:

```plaintext
BP #f:12d5 "J ax!=10;g"
BP #f:12d5 "J ax==10;t;g"
BP SchedNextRet "J wo(Tasknumber)==8,'.p*;.r';g"
BP DOSOPEN "J wo(Tasknumber)==32,'da #(wo(ss:sp+20)):(wo(ss:sp+1e));g';g"
```

The first example shows a breakpoint set at address #f:12d5. When this breakpoint fires the **J** command tests the condition **AX** register not equal to decimal 10. If this is true, the **G** command is executed. Since no **cmd2** is specified the **J** command returns control to the debugging console when the condition is false (**AX** equal to decimal 10).

The second example is has the same effect as the first but is implemented by testing the logically opposite condition.

The third example shows one method of stopping the system when a thread switch to a particular thread slot has just occurred. In this case the debugging console gains control when thread slot 8 is selected, whereupon **.p** and **.r** are automatically executed. The breakpoint **SchedNextRet** is one of two exit points from the scheduler (**tkSchedNext**). The other, **SchedNextRet2** is taken when the same thread slot is selected for re-dispatch. The global variable **Tasknumber** contains the current and therefore out-going slot number on entry to the scheduler; and in-coming slot number on exit from the dispatcher.

**Note:**

The kernel calls one of the **KMExitKmode** routines before giving control to user code. During this kernel exit processing the **Resched** and ( **TCB** and **PTDA** ) force flags are checked again and if set the scheduler/dispatcher sequence is invoked. It is possible therefore, that even though a thread is selected to run, and achieves **run** state, it is put back on the ready queue before being given any user processing time.

The fourth example illustrates a method of tracing resources that are opened by a specific thread slot (in this case slot 32) without giving control to the debugging console. **DOSOPEN** is the kernel's entry point for open processing. At this point words **0x0f** and **0x10** contain the offset and selector that points to the resource name.

-----------------------------

**K - Display Stack Trace from Address**
Display the stack-trace from a given stack frame address.

Syntax:

K
KS          stack-frame          selector:offset
KB

Parameters:

K
Display stack frame trace assuming the default operation size from the code descriptor specified by selector:offset.

KS
Display frame trace assuming an operation size of 16-bits (small-model).

KB
Display frame trace assuming an operation size of 32-bits (big-model).

stack-frame
Address of the starting stack-frame. If not specified then this defaults to the current SS:EBP or SS:BP as set by the last register display.

selector:offset
The selector:offset address of the code that is in effect when the starting Stack-frame address was created. If not specified this defaults to the current CS:EIP or CS:IP as displayed by the See the R command.

The code selector associated with this address is used for two purposes:

1. To determine the default operand size in effect from the code segment descriptor.
2. To attempt to distinguish between near and far calls at the starting stack-frame address.

Results & Notes:

The K command displays the stack trace, threading through the BP or EBP chain until either an invalid chain pointer is encountered or the command is interrupted by the user. For each stack-frame, the return address and for parameter words or double-words are displayed. The symbol associated with the return address is displayed after the parameter words. An example is given below:

```
##.S 8
##.R
eax=00d7e7f6 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
```

Notes
1. The K command is insensitive to unconventional use of the stack, such as where subroutine returns are effected explicitly by setting the stack pointer and jumping back to the calling code or in optimised code where the EBP or BP registers are not used as stack-frame pointers.

   Such possibilities exist within the system when for example the kernel returns to user code and also within some Presentation Manager components.

2. No attempt is made to trace correctly through thunking layers where the default operand size changes.

3. The stack trace is insensitive to any explicit segment operand overrides that may be active.

4. No attempt is made to examine the descriptor of the SS register to determine whether EBP or BP should be used. In much 32-bit code both the 16-bit and 32-bit data descriptors are created by the system for calls to 16-bit subroutines.

   In the example above the stack-frame address has been explicitly overridden to use BP since the 16-bit stack selector (1f) is in effect rather than the 32-bit SS selector.

5. Unlike the default stack-frame address the default code selector:offset is taken from the register values on entry to the Kernel Debugger.

Warning:

   In consequence of these points it recommended that the stack-frame and code selector:offset addresses be explicitly coded when using the K command, as in the example above. In addition the stack trace should be verified with a memory dump of the stack.

--------------------------------------------

L - List Maps, Groups and Symbols

List maps, groups and symbols from loaded symbol files.

See the W command for related information.

Syntax:

```
L       A                                                 `M          map-name
G                                               `S                                                 `addr
N                                                 `addr
symbol
```

Parameters:

A

List absolute symbol definitions for the specified map-name or for all active maps.

M

List all active maps or the status of the specified map.

G

List groups defined in all active maps of the specified map.
When \texttt{addr} is specified this option list the nearest \texttt{symbols} to the address. If an exact match is found that symbols is listed otherwise the nearest symbol before and after \texttt{addr} is listed.

When \texttt{symbol} is specified then the address, map and group corresponding to the symbol is listed.

If neither \texttt{addr} nor \texttt{symbol} is specified then the default disassembly address is assumed. See the \texttt{.R} and \texttt{U} commands for related information.

\textbf{S}

List all symbols defined in the group that encompasses \texttt{addr} for all active maps. If \texttt{addr} is not specified then the value of \texttt{CS:EIP} on entry to the debugger is assumed, as displayed by the \texttt{R} command.

\texttt{map-name}

Specifies the link edit map name from which information is to be displayed.

\texttt{addr}

Specifies an explicit \texttt{address expression}.

\texttt{symbol}

Specifies a publicly defined \texttt{symbol} name from a program source code.

\textbf{Results & Notes:}

Symbol maps are obtained from symbol files (*.SYM), which are generated using the \texttt{linkage editor} and the \texttt{MAPSYM} utility. Under the Kernel Debugger they are loaded from the same directory as their corresponding load module when that is loaded by the system. When this happens the \texttt{Symbols linked (map-name)} message appears. When a load module is deleted from the system, its map is removed and the message \texttt{Symbols unlinked (map-name)} appears.

Under the Dump Formatter symbol files are loaded for each \texttt{MTE} in the dump, during initialisation, from the current directory (usually the directory the Dump Formatter is running from).

Under the Dump Formatter conforming segments are not checked. Thus a ring 2 selector:offset address may not be recognised, whereas the ring 3 selector is. If \texttt{LN} does not find a symbol for a ring 2 selector, try specifying the same selector with the ring 3 RPL specified. For example, specify \texttt{d0fe:1234} as \texttt{d0ff:1234}.

Under the Dump Formatter \texttt{LN} does not check equivalences of the selector:offset and linear forms of an address. Therefore it may be necessary to apply the \texttt{CRMA} to an address if \texttt{LN} fails to find any near symbols.

Loaded symbol maps be \texttt{active} or \texttt{inactive}, depending on whether the corresponding load module is (potentially) active in the current context. In the case of private executable modules erroneous symbolic information may be associated with a private storage location. For this reason maps may be manually activated and removed using the \texttt{W} command.

Maps for WINOS2 and WINDOWS components are supported under the Kernel Debugger only. These are automatically activated and deactivated according to whether the Kernel Debugger default \texttt{thread slot} is a WINDOWS or WINOS2 environment.

Output from each of the \texttt{L} subcommands is more or less self explanatory. Examples follow:

\begin{verbatim}
##la
cmd: 9876 __acrtmsg
9876 __acrtused
d6d6 __aDBused
d6d6 __aDBdoswp

List absolute symbols defined in cmd.exe and their associated constants.
\end{verbatim}

\begin{verbatim}
##lm
cmd is active
kernel [0040, 003f]
minxobj is active
wprint is active
nwiapi is active
rexxnit is active
pmmle is active
fka is active
ibmdevr is active
ibmvgar is active
pmpre is active
\end{verbatim}
os2krnl is active

List current map status.

Note:

The Windows Kernel is not active, but loaded in thread slots 40 and 3f. The additional active slot number information is only provided with WINDOWS and WINOS2 environment map files.

##lg cmd
cmd:
000f:00000000 _TEXT1
0017:00000000 _TEXT3
001f:00000000 DGROUP

List segment groups defined in cmd.exe and their associated addresses.

##ln %20000
%00020000 cmd:_TEXT3:_eChcp
##ln _tkschednext
%fff4521c os2krnl:DOSHIGH32CODE:_tkSchedNext
##ln
0170:fff44695 os2krnl:DOSHIGH32CODE:HaltInst + 1
0170:fff44787 postSchedNext - f1

List near symbols and their associated addresses.

Note:

In this example three uses of LN are shown.

1. Address %20000 is shown to coincide with _eChcp in the _TEXT3 group of CMD.EXE
2. Symbol _tkschednext is shown to be at address %fff4521c in the DOSHIGH32CODE of OS2KRNL.
3. The current CS:EIP is at +1 byte from HaltInst in group DOSHIGH32CODE of module OS2KRNL and -f1 bytes before postSchedNext in the same group and module.

##ls %fff3f500
%fff3f4a4 DevWOHandle
%fff3f4ac g_CodeLockProc
%fff3f4b1 CodeLockProc
%fff3f5a4 g_CodeUnlockProc
%fff3f5a9 CodeUnlockProc
%fff3f614 _FSAbortVDM
%fff3f62c FS32IREAD
%fff3f638 FS32WRITE
%fff3f644 w_Big32I0
%fff3f6c0 w_SetFileLocks
%fff3f6c8 w_ProtectSetFileLocks

List symbols in the current group encompassing address %fff3f500

See the W command for related information.
M - Move a Block of Data in Memory

Move a block of contiguous data from one memory location to another. This command guarantees to duplicate the source data even when source and destination overlap.

Syntax:

```
M source-addr Ln target-addr
```

**Parameters:**

- `source-addr`: The source address of the memory location to be moved (copied).
  
  An address expression may be specified.

- `Ln`: The number (n) of bytes to move.

- `target-addr`: Target address of the memory move operation.
  
  An address expression may be specified.

**Results & Notes:**

Memory content is copied from the source to the target address. If the source and target overlap then source will be updated, however the move operation is conducted from highest to lowest address or *vice versa* depending on whether the target address is higher or lower than the source, thereby guarantee-ing a faithful copy of the original source.

If memory is present the storage is updated otherwise an *Invalid Address* message is generated. If this should happen, valid storage may be paged into memory by means of the .I command.

O - Output to an I/O Port

Output a byte of data to a 16-bit I/O Port

Syntax:

```
O port data
```

**Parameters:**

- `port`: 16-bit I/O port address

- `data`: A byte of data expressed numerically. This may be specified as a simple numeric expression.

**Results & Notes:**
The byte is sent to the requested I/O port.

Note:

This command may be used to set the debugging communication port parameters from the Kernel Debugger initialisation command file (KDB.INI) as in the following example:

```
Set COM2 DTR line (assume standard port assignment for COM2 that is, 2f8):
#O 2fc 1

Set COM1 DTR line (assume standard port assignment for COM1 that is, 3f8):
#O 3fc 1
```

--------------------------------------------

P - PTrace Instruction Execution

Trace instruction execution within a single procedure. This command is very similar to the T command, except that CALL, loop and string repeat instructions are traced as single instructions (even though allowed to execute correctly).

Syntax:

```
P N = start-addr count
```

Parameters:

(Default)

Trace instruction execution by single-stepping, treating CALL loop and string repeat instructions as single events.

Note:

Certain areas of the system are known to cause problems if traced. Attempts to trace these areas are intercepted by the Kernel Debugger. See below for further information.

N

Trace instructions but suppress the register display after each instruction is executed.

T

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the IDT and the system's re-instated until after then next instruction has executed. After execution of the next instruction the the Kernel Debugger's V commands are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the IDT, using a command sequence similar to:

```
VC n
P
VS n
```

start-addr

The address from which execution is to continue. This must be a valid address for the current context. If start-addr is omitted then execution continues from the current CS:EIP, as shown by the R command.

count

The number of instructions to trace before re-entering the Kernel Debugger unless one of the following conditions is
encountered:

- A fatal exception occurs
- An Internal Processing Error (IPE) occurs.
- A ‘sticky’ breakpoint fires
- A non-maskable interrupt occurs
- An INT 3 instruction is executed
- The user enters Ctrl-C from the debugging console.

If omitted then count defaults to 1 instruction.

Results & Notes:

The Ptrace commands trace the execution of machine instructions, and by default, display the current registers and next instruction to execute at each step. For the purposes of the displayed trace, the CALL instruction does not have the called routine traced, but tracing resumes on return. Loop and string repeat instructions are also treated as atomic entities with the instruction following the loop or repeat shown as the next to execute. INT 3 instructions are stepped over to avoid a double breakpoint at the same address even though they appear as the next instruction to execute.

The following system routines are known to causes inconsistency or even system failure if traced. Consequently Ptrace will suspend tracing until after execution leaves these routines.

```
_Debug_CtrIC32 through _EndCtrIC32
_DebugLoadSymMTE through EndDebugLoadSymMTE
_PGSwitchContext through pgSwitchRet
```

See the TX command for information on tracing these routines.

PN suppresses the register display from the automatic R command, but still displays an unassembled next instruction for each traced instruction. If the ZS command has been used to specify a different default command then PN behaves exactly as P.

An example of the output from PN is as follows:

```
##PN 5
0170:fff4521f 803d9e53e0ffff cmp byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4           jnz fff451dc
0170:fff45228 803d9643e0ff00 cmp byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be           jnz fff451ef
0170:fff45231 0f01e1         smsw cx
##
```

Note: The last traced instruction is the next to be executed.

Warning:

If any of the PTrace commands is interrupted, the Kernel Debugger may leave a temporary break-point active. This will result in a Trap 1 when the system is next given control. If this occurs then either of the PT or GT commands will clear this condition.

Q - Quit the Dump Formatter

Quit the Dump Formatter.

Syntax:
Parameters:
None

Results & Notes:
The Dump Formatter is terminated.

Warning:
Do not use this command when the Dump Formatter is invoked from PMDF. This will cause PMDF to hang. To terminate the Dump Formatter either quit PMDF from the system menu or select another dump for processing.

R - Set or Display Current CPU Registers

Display or set the current CPU registers saved on entry to the Kernel Debugger. Set default addresses for E command, D command, K command and U command.

Under the Dump Formatter this command is implemented as an alias to the .R command. Also applicable to the Dump Formatter only, the default addressing mode is not set according to the VM flags of the EFLAGS register but is assumed always to be in protect mode. This has been corrected from fix pack 29 of Wapr 3.0 and base Warp 4.0.

The remaining discussion in the section applies to the Kernel Debugger.

Syntax:

```
R
```

```
T
flag register
2-bit flag
16-bit register
32-bit register

flag register:
```

F
EF    flag mnemonics

CR0    cr0 flag mnemonics

MSW    msw flag mnemonics"
2-bit flag:
   IOPL
       pl

16-bit register:
   AX
   BX
   CX  16-bit value
   DX
   SI
   DI
   SP
   BP
   IP
   PC
   ES
   CS
   DS
   SS
   FS
   GS
   TR
   IDTL
   GDTL
   LDTR

32-bit register:
   EAX
   EBX
   ECX  32-bit value
   EDX
   ESI
   EDI
   ESP
   EBP
   EIP
   CR2
   CR3
   CR4
   DR0
   DR1
   DR2
   DR3
   DR6
   DR7
   TR6
   TR7
   IDTB
   GDTC

Parameters:

(Default)
Displays the current CPU registers on entry to the Kernel Debugger and sets default addresses for E command, D command and U command.

Register mnemonics are assigned the values displayed for use in address expressions and operands of other Kernel Debugger and Dump Formatter commands.

Note:

The .SS command may be used to change the displayed values of CS, EIP, SS and ESP. It does not affect the values restored then the Kernel Debugger returns control to the system.

T
Toggle register display mode between terse and non-terse forms. The terse form suppresses display of the test, debug, control, descriptor table and task registers.
This option affects both the `R` and `.R` commands.

**flag register**

Specifies one of the flag registers to be modified. The following mnemonics may be used:

- **F** 80286 `FLAGS` register.
- **EF** 80486 `EFLAGS` register.
- **MSW** Machine status word.
- **CR0** Control register 0

Each of the flag bits is specified by a mnemonic. More than one flag may be specified, order being unimportant. The Kernel Debugger processes the flags from left to right, if an invalid flag is encountered processing stops, but those flags already processed remain in effect.

Some flags are toggled by specifying a single mnemonic, others use a one mnemonic for the set condition and a another of the reset condition.

If replacements flags are omitted then the user is prompted for values.

**flag mnemonics**

Specifies one or more updated flags values for the `FLAGS` or `EFLAGS` registers.

The following mnemonics are defined. The value of `t` implies the flag value is toggled when the mnemonic is specified:

<table>
<thead>
<tr>
<th>Flag Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>17</td>
<td>Virtual 8086 Mode (EFLAGS only)</td>
</tr>
<tr>
<td>RF</td>
<td>16</td>
<td>Resume Flag – Disable Debug Exceptions (EFLAGS only)</td>
</tr>
<tr>
<td>NT</td>
<td>14</td>
<td>Nested Task</td>
</tr>
<tr>
<td>OV</td>
<td>11</td>
<td>Overflow</td>
</tr>
<tr>
<td>NV</td>
<td>11</td>
<td>¬Overflow</td>
</tr>
<tr>
<td>DN</td>
<td>10</td>
<td>Direction Down</td>
</tr>
<tr>
<td>UP</td>
<td>10</td>
<td>Direction Up</td>
</tr>
<tr>
<td>EI</td>
<td>9</td>
<td>Enable Interrupts</td>
</tr>
<tr>
<td>DI</td>
<td>9</td>
<td>Disable Interrupts</td>
</tr>
<tr>
<td>NG</td>
<td>7</td>
<td>Negative Sign</td>
</tr>
<tr>
<td>PL</td>
<td>7</td>
<td>Plus Sign</td>
</tr>
<tr>
<td>ZR</td>
<td>6</td>
<td>Zero Result</td>
</tr>
<tr>
<td>NZ</td>
<td>6</td>
<td>Non-zero Result</td>
</tr>
<tr>
<td>AC</td>
<td>4</td>
<td>Auxiliary Carry</td>
</tr>
<tr>
<td>NA</td>
<td>4</td>
<td>¬Auxiliary Carry</td>
</tr>
<tr>
<td>PE</td>
<td>2</td>
<td>Parity Even</td>
</tr>
<tr>
<td>PO</td>
<td>2</td>
<td>Parity Odd</td>
</tr>
<tr>
<td>CY</td>
<td>0</td>
<td>Carry</td>
</tr>
<tr>
<td>NC</td>
<td>0</td>
<td>¬Carry</td>
</tr>
</tbody>
</table>

**cr0 flag mnemonics**

Specifies one or more updated flags values for the `CR0` register.
The following mnemonics are defined:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG</td>
<td>31</td>
<td>1</td>
<td>Paging Enabled</td>
</tr>
<tr>
<td>ET</td>
<td>4</td>
<td>1</td>
<td>Extension Type Flag - x87 support</td>
</tr>
<tr>
<td>TS</td>
<td>3</td>
<td>1</td>
<td>Task Switch Flag</td>
</tr>
<tr>
<td>EM</td>
<td>2</td>
<td>1</td>
<td>Emulation exception</td>
</tr>
<tr>
<td>MP</td>
<td>1</td>
<td>1</td>
<td>Math Present</td>
</tr>
<tr>
<td>PM</td>
<td>0</td>
<td>1</td>
<td>Protect Mode Enabled</td>
</tr>
</tbody>
</table>

**msw flag mnemonics**

Specifies one or more updated flags values for the MSW register.

The following mnemonics are defined:

<table>
<thead>
<tr>
<th>flag</th>
<th>bit</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>3</td>
<td>1</td>
<td>Task Switch Flag</td>
</tr>
<tr>
<td>EM</td>
<td>2</td>
<td>1</td>
<td>Emulation exception</td>
</tr>
<tr>
<td>MP</td>
<td>1</td>
<td>1</td>
<td>Math Present</td>
</tr>
<tr>
<td>PM</td>
<td>0</td>
<td>1</td>
<td>Protect Mode Enabled</td>
</tr>
</tbody>
</table>

**2-bit flag**

This option is used to specify that the IOPL field of the FLAGS or EFLAGS register should be updated with the specified replacement **2-bit value**. The mnemonic IOPL is coded to specify this option.

If the replacement value is not specified then the user is prompted for a value.

**16-bit register**

This option is used to set the value of a register where **16-bit register** specifies either one of the standard INTEL register mnemonics or:

- **GDTL** The GDT limit.
- **IDTL** The IDT limit.
- **PC** The program counter. This is synonymous with IP.

This option implies a request to update a register value. If the corresponding new **16-bit value** is not specified then the prompted for a replacement value.

**32-bit register**

This option is used to set the value of a register where **32-bit register** specifies one of the standard INTEL register mnemonics or GDTB or IDTB.

This option implies a request to update a register value. If the corresponding new **32-bit value** is not specified then the prompted for a replacement value.

**2-bit value**

Specifies the 2-bit replacement value for the IOPL.

**16-bit value**

Specifies the 16-bit replacement value for a given 16-bit register.

**32-bit value**

Specifies the 32-bit replacement value for a given 32-bit register.

*Results & Notes:*
The register information is obtained from a special save area when the Kernel Debugger is entered and restored from this area when control returns to the system.

When no operands are specified the `R` command operates in display mode in exactly the same manner as the `.R` command.

From fix pack 29 for Warp 3.0 and base Warp 4.0 Pentium Processor support was added to the Kernel Debugger. This allows `CR4` to be specified as a register mnemonic, though `CR4` is never displayed without specifying it explicitly as an operand to the `R` command. On non-Pentium systems, `CR4` is shown as `00000000`.

When operands are specified the `R` command operates in alter mode. If no replacement value is supplied on the command then the user is prompted with the current value followed by a colon prompt character. For example:

```
#R SS
0030
```

Flag register value prompts have their current flag setting interpreted using the mnemonics described above. For example:

```
#R EF
--(rf) --(vm) --(nt) nv(ov) up(dn) ei(di) pl(ng) nz(zr) na(ac) po(pe) nc(cy)
```

This example shows mnemonics for current settings followed by their negating mnemonic in brackets. For example:

- **RF** is not in effect, but since it is a toggle flag, the value `RF` specified at the prompt would set `RF`.
- **NV** is in effect. To negate it, specify `OV` at the prompt.

--------------------------------------------

**S** - Search Memory for Data

`S` search a memory range for occurrences of a list of bytes.

**Syntax:**

```
S          addr       Ln               value              "string"
```

**Parameters:**

- **addr**
  The address of the memory location to be searched.

- **Ln**
  The number (n) of bytes to search.

- **value**
  A numerical byte value to be searched into memory. One or more values may be specified separated commas or blanks. These may be mixed with "string" values.
**string**

A character string enclosed in single or double quotes. Each character is treated as a list byte values to search memory, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical **values**.

**Results & Notes:**

The list of **values** and **strings** is used as a combined search argument. Only precise matches against the entire search argument are reported. The search is repeated for every byte location in the range specified. If no matches are found then nothing is displayed. Where matches are found the **Search** command displays a list of storage addresses. For example:

```
## s ptda_start l1000 "TD"
0030:0000fffe
ln 30:fffe
0030:0000fffe os2krnl:TASKAREA:ptda_signature
```

If memory is present the storage is updated otherwise an **Invalid Address** message is generated. If this should happen, valid storage may be paged into memory by means of the `.I` command.

```
```

---

**T - Trace Instruction Execution**

Trace instruction execution singly or for a specific number or instructions or to a specific address.

**Syntax:**

\[
\begin{align*}
T & = \text{start-addr} & \text{count} \\
TX & = \text{start-addr} & \text{count} \\
TN & = \text{start-addr} \\
TT & = \text{start-addr} \\
TA & = \text{start-addr} & \text{break-addr} \\
TC & = \text{start-addr} \\
TS & = \text{start-addr} & \text{break-addr}
\end{align*}
\]

**Parameters:**

*(Default)*

- **T**
  - Trace one or more instructions, excluding known bad areas (see X subcommand below).
- **A**
  - Trace all instructions to **break-addr**.
  - This option requires **break-addr** to be specified.
- **C**
  - Counts all instructions executed until **break-addr** is reached.
  - **Note:** Counting is suspended when the system switches out of the current context in which the **TC** command was executed. It is resumed when that context switches back.
  - This option requires **break-addr** to be specified.
- **N**
  - Trace instructions but suppress the register display after each instruction is executed.
- **S**
  - The trace special option is similar to **TC** except that an intermediate instruction count is displayed before execution of each **CALL** instruction and after each return.
This option requires \texttt{break-addr} to be specified.

\section*{Notes}

Counting is suspended when the system switches out of the current \texttt{context} in which the \texttt{TS} command was executed. It is resumed when that context switches back.

\texttt{TS} does not attempt to match \texttt{CALL} with \texttt{RET} instructions. Instead it inserts a temporary \texttt{breakpoint} at the instruction address following the \texttt{CALL}. In addition \texttt{TS} maintains a 'stack' of return addresses and always checks the most recent two entries, as it single-instruction steps through the traced code, for a matching return address. This technique enables code that uses \texttt{JMP} instructions to return from a call to be better detected. This is not a foolproof technique, especially where mutually recursive code is traced.

\section*{T}

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the \texttt{IDT} and the system's re-instated until after then next instruction has executed. After execution of the next instruction the the Kernel Debugger's \texttt{V commands} are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the \texttt{IDT}. using a command sequence similar to:

\begin{verbatim}
VC n
T
VS n
\end{verbatim}

\section*{X}

This option forces the Kernel Debugger to trace areas of system code that are known to be unsuitable for tracing. Normally, when \texttt{Trace} encounters one of the following routines:

\begin{verbatim}
_Debug_Ctr1C32 through _EndCtrlC32
_DebugLoadSymMTE through EndDebugLoadSymMTE
_PGSwitchContext through pgSwitchRet
\end{verbatim}

a temporary breakpoint is inserted at the routine's return address and the system is allowed to go to that address uninterrupted. When \texttt{TX} is used the Kernel Debugger will attempt to trace instructions within these routines.

The consequence of forcing tracing in these routines may be at worst, the system is left in an unrecoverable state, and at best certain Kernel Debugger commands will give erroneous information.

\section*{start-addr}

The address from which execution is to continue. This must be a valid address for the current context. If \texttt{start-addr} is omitted then execution continues from the current \texttt{CS:EIP}, as shown by the \texttt{R command}.

\textbf{Warning:} Be very careful to ensure that the start address is valid for the privileged level and addressability of the code and data selectors in use. If the Kernel Debugger attempts to load a segment register that is invalid the system may trap in the debugger code.

\section*{break-addr}

The address at which tracing will stop and the Kernel Debugger will be re-entered unless one of the following conditions is encountered:

\begin{itemize}
  \item A fatal exception occurs
  \item An Internal Processing Error (IPE) occurs.
  \item A 'sticky' breakpoint fires
  \item A non-maskable interrupt occurs
  \item An \texttt{INT 3} instruction is executed
\end{itemize}

The user enters \texttt{Ctrl-C} from the debugging console. The \texttt{break-addr} only remains in effect until the Kernel Debugger is next re-entered.

\section*{count}

The number of instructions to trace before re-entering the Kernel Debugger unless one of the following conditions is encountered:

\begin{itemize}
  \item A fatal exception occurs
\end{itemize}
An Internal Processing Error (IPE) occurs.

A 'sticky' breakpoint fires

A non-maskable interrupt occurs

An INT 3 instruction is executed

The user enters Ctrl-C from the debugging console.

If omitted then count defaults to 1 instruction.

Results & Notes:

Except for TN, TC and TS the default command is executed when control returns to the debugging console. This defaults to the R command unless respecified through use of the ZS command.

TN suppresses the register display from the automatic R command, but still displays an unassigned next instruction for each traced instruction. If the ZS command has been used to specify a different default command then TN behaves exactly as T.

An example of the output from TN is as follows:

```plaintext
##TN 5
0170:fff4521f 803d9e53e0ffff cmp byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4           jnz     fff451dc
0170:fff45228 803d9643e0ff00 cmp byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be           jnz     fff451ef
0170:fff45231 0f01e1         smsw    cx
##
```

Note: The last traced instruction is the next to be executed.

TC displays the total number of instructions trace in the following message:

```
Total traced instructions: nnnn (decimal)
```

where nnnn is the number of traced instructions.

Following this message the default command is executed. See the Z command for details.

TS displays a variety of different messages, examples of which are:

```
----------------------------------------------------------------------
Instruction Count: 101
d0df:0000f319 9a0000c810     call    10c8:0000
----------------------------------------------------------------------
Accumulated number of instructions executed before the CALL instruction.

Accumulated number of instructions executed when the return address is encountered.

Note: This does not include the instruction at the return address.

----------------------------------------------------------------------
...Special exit follows...
Exit: 360
```

Accumulated number of instructions executed when the second most recent return address is encountered. In this case the most recent return address is discarded from the 'stack'.

Note: This does not include the instruction at the return address.
Switching context...
...Back in context

Signifies context switching occurring and the suspension and resumption of instruction counting.

Total traced instructions: nnnn (decimal)

The total number of instructions traced when the `break-addr` is encountered.

Notes

- **REP** and **REPNE** string instruction prefixes are handled differently to other instructions when single stepping. The Kernel Debugger generates a temporary break-point following the repeated string instructions (MOVS, CMPS, SCAS, LODS and STOS) and returns control to the system until the temporary break-point fires.

- **INT 3** instructions encountered when single-stepping are reported but in actual fact stepped over, thereby avoiding a double break-point at the same address.

Warning:

If any of the Trace commands is interrupted, the Kernel Debugger may leave a temporary break-point active. This will result in a Trap 1 when the system is next given control. If this occurs then either of the **TT** or **GT** commands will clear this condition.

U - Unassemble

Unassemble storage at a given address.

Syntax:

```
U addr
```

Parameters:

- `addr`
  
  The address of the storage location to be unassembled.

Results & Notes:

The `U` command unassembles storage from the address given. No attempt is made to distinguish between code and data storage. If no `addr` is given then the default address is determined in order of precedence as follows:

- The last unassembled address + 1, or
- The `CS:EIP` of the last explicitly executed .R command or R command, or
- The address of the next instruction to be executed.

The `U` command takes its default addressing mode as set by the `R` or .R commands. Prior to fix pack 29 for Warp 3.0 and base Warp 4.0 V8086 addressing mode was ignored by the Dump Formatter unless explicitly specified by using the & addressing operator.

Output from the `U` command is in two forms depending on whether the storage address was set in the context of the default (Kernel...
Debugger’s or Dump Formatter’s current thread slot or another slot. In the former case output appears as in the following example:

```
#u
0170:fff4521f 803d9e53e0fff cmp byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4 jnz fff451dc
0170:fff45228 803d9643e0ff0 cmp byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be jnz fff451ef
0170:fff45231 0f01e1 smsw cx
0170:fff45234 66f7c10200 test cx,0002
0170:fff45239 0f8552050000 jnz fff45791
0170:fff4523f fa cli
```

In the latter case the context is shown by prefixing the thread slot to the address as in the following example:

```
##p*
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD     pPTDA    pTCB     Disp SG Name
0022# 0013 0003 0013 0001 blk 0300 7b6ea000 7b8c7128 7b8ab820 1eb8 18 epm
##r 34
eax=00000000 ebx=000007f4 ecx=00000000 edx=0003ace7 esi=d02f4ef0 edi=000011ec
eip=0000272d esp=0000755e ebp=00007566 iopl=2 -- ---- nv up ei ng nz na pe nc
 cs=d02f ss=001f ds=bccf es=ace7 fs=150b gs=000  cr2=15b20000  cr3=001d9000
doscall1:CONFORM16:postDOSSEMWAIT:
0034|d02f:0000272d c9 leave
```

Note:
The unassembled instruction mnemonics may be toggled between upper and lower case by use of the Y command.

```
```

V - Exception/Trap/Fault Vector Commands

This group of commands manipulates IDT entries 0 through e to point to Kernel Debugger supplied interrupt handlers. By this means the Kernel Debugger may selectively be made to intercept each system exception before the system is allowed to process the exception. When a system exception is intercepted the Kernel Debugger gives control to the user. The original IDT entries are retained so that they may be re-instated, or given control following an exception which the Kernel Debugger has been intercepted. See the GT and TT commands for information in returning control to the system exception handlers.

Syntax:

```
V L S R interrupt
T V C P
```
Parameters:

L
The List subcommand lists active Kernel Debugger trap and interrupt vectors.

Only a category specification (R, V, P, F or N) may be optionally specified with the List subcommand.

S
The Set subcommand activates a Kernel Debugger exception vector according to criteria specified in the remaining parameters. Vectors set using this option cause the Kernel Debugger to receive control only when the corresponding exceptions are generated in ring 2 and 3 code.

T
The Trap subcommand activates a Kernel Debugger exception vector according to criteria specified in the remaining parameters. Vectors set using this option cause the Kernel Debugger to receive control whenever the corresponding exceptions are generated regardless of the current privileged level.

C
The Clear subcommand re-instates one or more system exception handlers according to the criteria specified in the remaining parameters.

R
This option refines the exception criteria to Real-mode exceptions only.

If no refining category is specified then the vector subcommand being executed applies to the R, V, P and F options simultaneously.

V
This option refines the exception criteria to V86-mode exceptions only.

P
This option refines the exception criteria to Protect-mode exceptions only.

F
This option refines the exception criteria to those exceptions that would be 'fatal' to a process or the system. If a Local (system) Exception handler is registered then the exception is not intercepted. User Exception Handlers do not affect the operation of the Vector command. Local Exception Handlers protect the system from recoverable errors, in particular bad parameters passed in API calls. If a parameter causes the system to trap, the Local Exception Handler is given control and the application is terminated. VSF will not intercept such traps. For further information on exception handling and how to intercept exceptions in general, see Trap and Exception Processing.

U
This option allows exceptions, fatal to a process, to be intercepted before the process is terminated. Interception occurs if the exception is not recovered by any user exception handler. VSU will intercept 'user-fatal' exceptions whether they originate from the system or user. In particular if a local (system) exception handler has intercepted a kernel trap due to a bad API parameter then this will be intercepted if no user exception handler recovers from the error.

Note: This option was introduced with OS/2 Warp V3.0 fix pack 26 and OS/2 Warp V4.0 fix pack 1.

N
This option causes the Kernel Debugger exception handler to 'beep' continuously instead of giving control to the user. The user may then break into the Kernel Debugger by entering Ctrl-C at the debugging console.

The N option works in conjunction with the four refining categories, that is, it does not by itself cause an interrupt to be trapped but instead specifies an action when that event occurs.

The N option must be explicitly specified for all four subcommands (L, S, T and C) when required.

interrupt
This allows one or more interrupt vectors, separated by commas, to be specified with the vector command as a refining criterion.

It is not valid with the List subcommand.

The abbreviation * may be specified as an alternative to the following interrupts, in each of the refining categories:
Results & Notes:

Only the List subcommand gives immediate output, which is of the form in the following example:

```
#VL
R 0 1 2 3 4 5 6
V
P d
U e
F e d
```

As can be seen from this example each category is shown with its one-letter abbreviation followed by a list of interrupt vectors currently being intercepted by the Kernel Debugger

Notes:

The N option must be specified explicitly to be listed.

The U option is only from OS/2 Warp V3.0 fix pack 26 and OS/2 Warp V4.0 fix pack 1.

All other subcommands only cause output to appear when an interrupt is intercepted. When this happens the following events occur:

1. The N option is checked, if enabled the Kernel Debugger emits a continuous beep until the user breaks in through the debugging console.
2. A trap message is issued if the default command is set to the R command.
3. The default command is executed.

The following figure shows the format of the trap messages issued by the Kernel Debugger exception handlers:

- Trap 0 - Divide Error Exception
- Trap 1 - Unexpected trace interrupt
- Trap 2 - NMI Interrupt
- Trap 4 - INTO Detected Overflow Exception
- Trap 5 - BOUND Range Exceeded Exception
- Trap 6 - Invalid Opcode Exception
- Trap 7 - Processor Extension Not Available Exception
- Trap 8 - Double Exception Detected nnnn
- Trap 9 - Processor Extension Segment Overrun
- Trap 10 (0AH) - Invalid TSS nnnn, mmmmmmm
- Trap 11 (0BH) - Segment Not Present nnnn, mmmmmmm
- Trap 12 (0CH) - Stack Segment Overrun or Not Present nnnn, mmmmmmm
- Trap 13 (0DH) - General Protection Fault nnnn, mmmmmmm
- Trap 14 (0EH) - Page Fault nnnn, mmmmmmm

In the messages above **nnnn** is substituted with the Intel exception code and **mmmmmmmm** is substituted with an interpretation of the Intel error code flags. For **Trap 10**, **Trap 11**, **Trap 12** and **Trap 13** the error code flags are interpreted as:

---

**External**  **External event**
IDT Gate  IDT gate selector error
GDT    GDT selector error
LDT    LDT Selector error

For Trap 14 the error code flags are interpreted as a combination of:

- Not Present  Page not present
- Read Access  Read Access failure
- Write Access Write Access Failure
- User mode    Fault occurred when executing in User mode
- Supervisor   Fault occurred when executing in Supervisor mode

If a trap occurs in the debugger component of the Kernel Debugger the trap message will be appended with:

- In Debugger

If this happens then the only hope of recovering the system is to set the registers, using the R command, to a known consistent set of values.

See the INTEL x86 Programmer’s Reference or the INTEL Pentium User’s Guide for further information on exceptions and error codes.

Notes

- Trap 1 normally occurs as part of the operation of the Kernel Debugger. Only unexpected Trap 1 exception are therefore reported.

- When a Trap 1 is generated through use of the Debug Registers then the Kernel Debugger signals this with the message Debug register hit.

- Trap 3 occurs through use of the INT 3 instruction. This is used both by the Kernel Debugger and user programs in implementing break-points. User programs may use the INT 3 instruction as a program controlled technique for breaking into the debugger. In these cases a trap message is not displayed.

W - Withmap Add/Remove

Add or remove a symbol map. Under the Kernel Debugger this merely activates or deactivates a symbol map. Under the Dump Formatter a symbol file may be re-loaded using the Withmap command.

Syntax:

```
WA 
WR map-name symbol-file *
```

Parameters:

- A

  Activate 1 or all symbol maps.

  **Note:** If the corresponding load module is not active then the map will remain deactivated. See the L command for more information on displaying map status.
R Remove 1 or all symbol maps.

L (not shown) This subcommand applied only to the Dump Formatter and has been superseded by the LM command.

map-name The symbol map name to be activated or deactivated

symbol-file The symbol file name, with optional path and extension, to be loaded or removed.

Note: This operand applied only to the Dump Formatter

* Specifies all maps or symbol files should be loaded or removed.

Results & Notes:
An error message is displayed only if the specified map-name is not loaded.
See the L command for related information.

Y - Set or Display Dump Formatter and Kernel Debugger Options

Set or display Dump Formatter and Kernel Debugger disassembly and register options.

Syntax:
Y?

Y

386ENV
REGTERSE
DIISR

Parameters:
(Default) Display current option settings.

? Display help for the Y command.

386ENV Force the Kernel Debugger and Dump Formatter to toggle the environment setting between 286 and 386 modes.

This affects the way in which commands interpret the register set. For example, in 286 mode, general registers are assumed, by default to be 16-bit registers. Under rare circumstances it necessary to force a particular mode to obtain a correct disassembly listing from the U command. Mostly this occurs in system code that is multi-modal and has juxtaposed sections of 32-bit and 16-bit code.

The initial setting is 386 mode under OS/2 V2.0 and above.

REGTERSE This has the same effect as the RT command.
The initial setting is for terse register display.

**DISLWR**

This option toggles upper and lower case display of assembler mnemonics from the U command. The initial setting is for lower case mnemonics.

### Results & Notes:

No information is displayed when setting options.

When querying options those in effect are displayed, for example:

```
#y
386env dislwr regterse
```

This shows that 386 environment is assumed, lower case disassembly is in effect and terse register display is active. If any one of these setting is toggled then the corresponding flag is not displayed.

---

**Z - Set, Execute or Display the Default Command**

Set, execute or display the default command.

**Syntax:**

```
Z                                                         L
S              cmd                                 "                  "
```

**Parameters:**

/(Default) Execute the default command string.

L Display the default command string

S Set the default command string.

`cmd` Specifies a Dump Formatter or Kernel Debugger commands to be use in the default command string. If the command string comprises more than one command the each must be separated by commas and the entire string enclosed in quotes.

### Results & Notes:

The default command string is executed automatically at breakpoints (where no other command string is associated with the breakpoint), after instruction tracing or when exception vectors are trapped. See the following commands for more information:

- B commands
- G command
P command
T commands
V commands

When the Kernel Debugger and Dump Formatter are initialised the default command string is set to "R".

Note:

When the user breaks into the Kernel Debugger with Ctrl-C the R command is executed regardless of the default command setting.

--------------------------------------------

External Commands

The following comprise the set of external commands:

.?. Display external command help
.A Display the SAS structure
.B Set COM Parameters
.C Display the Common A BIOS Data Area
.D Display an OS/2 System Structure
.H Display Dump Data Set Information
.I (KDB) Swap in Storage
.I (DF) Display Dump State
.K Display Ring 3 stack
.LM Format Loader structures (MTE, OTE, STE)
.M Format Memory Structures
.MA Format Memory Arena records (VMAR)
.MC Format Memory Context Records (VMCO)
.MK Format Memory Lock Information Records (VMLI)
.ML Format Memory Alias Records (VMAL)
.MO Format Memory Object Records (VMOB)
.MP Format Memory Physical Page Frame Tables
.MV Format Memory Virtual Frame Tables
.N Display Dump Header Information
.O Override default behaviour
.P Display Process and Thread Status Information
.PB Display Blocked Process Information
.PQ Display Scheduler Thread Queuing Information
.PU Display Process and Thread User Space Information
.R Display ring 2/3 registers
.REBOOT       Reboot the system under test
.S           Switch default thread slot
.SYSDUMP       Force a System Dump and Restart the System
.T           Format the System Trace Buffer

--------------------------------------------

.?- Show External Command Help

Display help for internal Kernel Debugger and Dump Formatter commands.

Syntax:

.?

Parameters: None.

Results & Notes:

Displays a help summary for most of the Dump Formatter and Kernel Debugger external commands.

Note:

Prior to fix pack 29 for Warp V3, some of the help information displayed is out-of-date. This information has been updated from fix pack 29 of Warp V3 and base Warp V4.

Two pages of information are displayed with an intervening --More-- prompt.

--------------------------------------------

.A - Format the System Anchor Segment (SAS)

Format the System Anchor Segment (SAS).

Syntax:

.A

Parameters:
none

Results & Notes:
The SAS is located from either GTD selector 70 or 78.

.A displays the following information:
--- SAS Base Section ---

SAS signature: SAS

offset to tables section: 0016
FLAT selector for kernel data: 0168
offset to configuration section: 001E
offset to device driver section: 0020
offset to Virtual Memory section: 002C
offset to Tasking section: 005C
offset to RAS section: 006E
offset to File System section: 0074
offset to infoseg section: 0080

--- SAS Protected Modes Tables Section ---

selector for GDT: 0008
selector for LDT: 0000
selector for IDT: 0018
selector for GDTPOOL: 0100

--- SAS Device Driver Section ---

offset for the first bimodal dd: 0CB9
offset for the first real mode dd: 0000
sel for Drive Parameter Block: 04C8
seg for ABIOS prot. mode CDA: 0000
seg for ABIOS real mode CDA: 0000
selector for FSC: 00C8

--- SAS Task Section ---

selector for current PTDA: 0030
FLAT offset for process tree head: FFF10910
FLAT address for TCB address array: FFF06BB6
offset for current TCB number: FFDFFB5E
offset for ThreadCount: FFDFFB62

--- SAS File System Section ---

handle to MFT Ptree: FE72CFBC
selector for System File Table: 00C0
sel. for Volume Parameter Bloc: 0788
sel. for Current Directory Struc: 07B8
selector for buffer segment: 00A8

--- SAS Information Segment Section ---

selector for global info seg: 0428
address of curtask local infoseg: 03C80000
address of DOS task’s infoseg: FFFFFFFF
selector for Codepage Data: 07CB

--- SAS RAS Section ---

selector for System Trace Data Area: 04B0
segment for System Trace Data Area: 04B0
offset for trace event mask: 0B28

--- SAS Configuration Section ---

offset for Device Config. Table: 0D50

--- SAS Virtual Memory Mgt. Section ---

Flat offset of arena records: FFF13304
Flat offset of object records: FFF1331C
Flat offset of kernel mte records: FFF0A891
Flat offset of context records: FFF1330C
Flat offset of linked mte list: FFF07934
Flat offset of page frame table: FFF11A70
Flat offset of page range table: FFF1118C
Flat offset of swap frame array: FFF03BAC
Flat offset of Idle Head: FFF10090
Flat offset of Free Head: FFF10080
Flat offset of Heap Array: FFF11B78
Flat offset of all mte records: FFF12E04

--- SAS Base Section ---

Marks the beginning of the SAS header section.

SAS signature

SAS signature from SAS_signature (SAS+0x0). Always set to character value “SAS”.

offset to tables section

Offset from SAS selector to the protected mode tables section.

Taken from SAS_tables_data (SAS+0x4).
FLAT selector for kernel data
Selector for 4G Read/Write addressability.
Taken from SAS_flat_sel (SAS+0x6).

offset to configuration section
Offset from SAS selector to the configuration tables section.
Taken from SAS_config_data (SAS+0x8).

offset to device driver section
Offset from SAS selector to the device driver section.
Taken from SAS_dd_data (SAS+0xa).

offset to Virtual Memory section
Offset from SAS selector to the Virtual Memory section.
Taken from SAS_vm_data (SAS+0xc).

offset to Tasking section
Offset from SAS selector to the Tasking section.
Taken from SAS_task_data (SAS+0xe).

offset to RAS section
Offset from SAS selector to the RAS data section.
Taken from SAS_RAS_data (SAS+0x10).

offset to File System section
Offset from SAS selector to the File System section.
Taken from SAS_file_data (SAS+0x12).

offset to infoseg section
Offset from SAS selector to the Infoseg section.
Taken from SAS_info_data (SAS+0x1e).

--- SAS Protected Modes Tables Section --
Marks the beginning of the protected mode tables section.

selector for GDT
GDT selector that maps the GDT.
Taken from SAS_tbl_GDT (SAS_tables_section+0x0).

selector for LDT
No longer used.
Taken from SAS_tbl_LDT (SAS_tables_section+0x2).

selector for IDT
GDT selector that maps the IDT.
Taken from SAS_tbl_IDT (SAS_tables_section+0x4).

selector for GDTPool
First GDT selector in selector pool. i.e. first non-predefined GDT selector.
Taken from SAS_tbl_GDTPool (SAS_tables_section+0x6).

--- SAS Device Driver Section --
Marks the beginning of the Device Driver Section.

offset for the first bimodal dd
Offset from SAS selector to the first device driver header in the device driver chain.
See .D command for formatting device driver headers.
Taken from SAS_dd_bimodal_chain (SAS_dd_section+0x0).
offset for the first real mode dd
No longer used.
Taken from SAS_dd_real_chain (SAS_dd_section+0x2).

sel for Drive Parameter Block
Selector that points to the head of the DPB chain.
See .D command for formatting DPBs.
Taken from SAS_dd_DPB_segment (SAS_dd_section+0x4).

sel for ABOIS prot. mode CDA
Selector for ABOIS protect mode CDA.
See .C command for displaying CDA information.
Taken from SAS_dd_CDA_anchor_p (SAS_dd_section+0x6).

seg for ABOIS real mode CDA
Segment for ABOIS real mode CDA. See .C command for displaying CDA information.
Taken from SAS_dd_CDA_anchor_r (SAS_dd_section+0x8).

selector for FSC
Selector for the FSC segment.
Taken from SAS_dd_FSC (SAS_dd_section+0x2).

--- SAS Task Section --
Marks the beginning of the tasking section

selector for current PTDA
Selector for the current PTDA and ring 0 stack.
Taken from SAS_task_PTDA (SAS_task_section+0x0).

FLAT offset for process tree head
Linear address of _pPTD AF irst, which contains the linear address of the PTDA that heads the PTDA tree.
Taken from SAS_task_ptdaptrs (SAS_task_section+0x2).

FLAT address for TCB address array
Linear address of _papTCBSlots, which contains the linear address of the TCB array.
Taken from SAS_task_threadaptrs (SAS_task_section+0x6).

offset for current TCB number
Linear address of _TaskNumber, which contains the current thread slot number.
Taken from SAS_task_tasknumber (SAS_task_section+0xa).

offset for ThreadCount
Linear address of _ThreadCount, which contains the highest thread slot number in use - 1.
Taken from SAS_task_threadcount (SAS_task_section+0xe).

--- SAS File System Section --
Marks the beginning of the File System Section

handle to MFT PTree
Linear address of the head of the Ptree for the .
See .D command for formatting MPTs.
Taken from SAS_file_MFT (SAS_file_section+0x0).

selector for System File Table
Selector for the segment containing a table of selectors that point to tables of SFTs. Each SFT table contains an 8 byte header followed by contiguous SFT entries.
See .D command for formatting SFTs.
sel. for Volume Parameter Bloc
This is the selector for the work buffer used by volume mount processing.
Taken from SAS_file_VPB (SAS_file_section+0x6).

Note:
The selector for the VPB segment is not given by this field. It may be located from the selector named by global variable GDT_VPB See .D command for formatting VPBs.

sel. for Current Directory Struc
Selector for the RMP segment containing CDS structures.
See .D command for formatting CDSs.
Taken from SAS_file_CDS (SAS_file_section+0x8).

selector for buffer segment
Selector for the file system buffer segment.
Taken from SAS_file_buffers (SAS_file_section+0xa).

--- SAS Information Segment Section --
Marks the beginning of the Information Section.

selector for global info seg
Selector for the Global Information Segment (GISEG).
Taken from SAS_info_global (SAS_info_section+0x0).

address of curtask local infoseg
16:16 far pointer for the current Local Information Segment (LISEG).
Taken from SAS_info_global (SAS_info_section+0x2).

address of DOS task's infoseg
Real mode local information segment pointer (unused).
Taken from SAS_info_localRM (SAS_info_section+0x6).

selector for Codepage Data
Selector for the segment containing the Code Page Data Information Block (CDIB).
Taken from SAS_info_CDIB (SAS_info_section+0xa).

--- SAS RAS Section --
Marks the beginning of the RAS section

selector for System Trace Data Area
Selector for the STDA trace buffer.
Taken from SAS_RAS_STDA_p
See .T command for formatting the system trace buffer. (SAS_RAS_section+0x0).

segment for System Trace Data Area
Selector for the STDA trace buffer.
Taken from SAS_RAS_STDA_r (SAS_RAS_section+0x2).
The same value is stored in both SAS_RAS_STDA_p and SAS_RAS_STDA_r.

offset for trace event mask
Offset from the SAS to the trace major event codes table (ras_mec_table).
Taken from SAS_RAS_event_mask (SAS_RAS_section+0x4).

--- SAS Configuration Section --
Marks the beginning of the Configuration section

offset for Device Config. Table
Offset from the SAS to the device configuration table.
Taken from SAS_config_table (SAS_config_section+0x0).

--- SAS Virtual Memory Mgt. Section --
Marks the beginning of the Virtual Memory Management section

Flat offset of arena records
The linear address of _parvmOne, the linear address of the first VM arena record (VMAR).
See .MA command for related information.
Taken from SAS_vm_arena (SAS_vm_section+0x0).

Flat offset of object records
The linear address of _pobvmOne, the linear address of the first VM object record (VMOB).
See .MO command for related information.
Taken from SAS_vm_object (SAS_vm_section+0x4).

Flat offset of context records
The linear address of _pcovmOne, the linear address of the first VM context record (VMCO).
See .MC command for related information.
Taken from SAS_vm_context (SAS_vm_section+0x8).

Flat offset of kernel mte records
The linear address of _DosModMTE, the kernel (DOSCALLS.DLL) MTE.
See .LM command for related information.
Taken from SAS_vm_krnl_mte (SAS_vm_section+0xc).

Flat offset of linked mte list
The linear address of _global_h, the linear address of the head of the MTE chain of link library modules.
See .LM command for related information.
Taken from SAS_vm_glbl_mte (SAS_vm_section+0x10).

Flat offset of page frame table
The linear address of _pft, the linear address of the first (frame 0) page frame structure (PF).
See .MP command for related information.
Taken from SAS_vm_pft (SAS_vm_section+0x14).

Flat offset of page range table
The linear address of _pgPageablePAI, the pageable PAI. The first double word of the PAI points to the page range table.
Taken from SAS_vm_prt (SAS_vm_section+0x18).

Flat offset of swap frame array
The linear address of _smbmDF, the linear address of swap frame allocation bit map followed by _smFileSize, the swap file size word length value in pages.
Taken from SAS_vm_swap (SAS_vm_section+0x1c).

Flat offset of Idle Head
The linear address of _pgIdleList, which points to the pseudo-PF at the head of the idle PF list.
See Idle Page Frame Structures for more information locating Idle Page Frame Structures.
See .MP command for related information.
Taken from SAS_vm_idle_head (SAS_vm_section+0x20).

Flat offset of Free Head
The linear address of _pgFreeList, which points to the pseudo-PF at the head of the free PF list.
See Free Page Frame Structures for more information locating Free Page Frame Structures.

See .MP command for related information.

Taken from SAS_vm_free_head (SAS_vm_section+0x24).

Flat offset of Heap Array

The linear address of _apkh, the array of VMKH kernel heap header structures. Note: the first entry is unused.

Taken from SAS_vm_heap_info (SAS_vm_section+0x28).

Flat offset of all mte records

The linear address of _mte_h, which is the linear address of the head of the MTE chain.

See .LM command for related information.

Taken from SAS_vm_all_mte (SAS_vm_section+0x2c).

--------------------

.B - Select the Communications Port and Speed

Select the communications port and speed.

Syntax:

.B           speed                                        ` port

Parameters:

speed

The COMx port speed. Any of the following values are valid:

150t
300t
600t
1200t
2400t
4800t
9600t
19200t

Note: Since baud rates are usually expressed in decimal and the default number base for the Kernel Debugger is hexadecimal then a t subscript must be supplied when using decimal values.

port

Specifies which COM port is to be used. If 1 or 2 are specified then COM1 or COM2 are implied. Any other numeric value is assumed to be an I/O port address.

Results & Notes:

When the Kernel Debugger initialises a default baud rate of 9600t is set.

The COM port defaults to COM2 if there are two serial ports otherwise COM1 unless no comports are defined in the ROM BIOS data area,
in which case the first port address in the ROM BIOS data area is assumed.

The parity, data and stop bit settings default to none, 8 and 1. These may be altered either:

- from the Kernel Debugger by writing directly to the `COM` port control register using the `O` command
- or from the system under test by using the `MODE` command.

If synchronisation is lost with the debugging console, for example because the debugging communications port has been temporarily used by another application then it may be reset using the `MODE` command from the command line of the system under test. For example, to re-specify the default parameters use:

```
MODE COM2 9600,n,8,1
```

---

**.C - Display the Common ABIOS Data Area**

Display ABIOS Command Data Area information.

**Syntax:**

```
.C
```

**Parameters:** None

**Results & Notes:**

`.C` displays data for each logical device ID anchored from the Common ABIOS Data Area (CDA). If the ABIOS is not present or initialised then the following message is displayed:

```
ABIOS Not Present or Not Initialised
```

The presence of ABIOS is indicated by a non-zero byte value located at symbol:

```
ABIOS_Present.
```

If the ABIOS is present and initialised then data based on the Logical Device ID (LID) table is displayed. The LID Table is located from a selector located at:

```
ABIOS_CDS_ANCHOR_p - in protect mode, or
ABIOS_CDS_ANCHOR_r - in real mode
```

Tabular data of the following form is displayed:

<table>
<thead>
<tr>
<th>LID (0000)</th>
<th>Type=Reserved</th>
<th>DB=001e:0114</th>
<th>FTT=0000:0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LID (0001)</td>
<td>Type=Null</td>
<td>DB=0000:0000</td>
<td>FTT=0000:0000</td>
</tr>
<tr>
<td>LID (0002)</td>
<td>Type=Internal</td>
<td>DB=0438:06f0</td>
<td>FTT=0448:011c</td>
</tr>
<tr>
<td>LID (0003)</td>
<td>Type=Diskette</td>
<td>DB=0438:0728</td>
<td>FTT=0448:012c</td>
</tr>
<tr>
<td>LID (0004)</td>
<td>Type=Video</td>
<td>DB=0438:07a4</td>
<td>FTT=0448:017c</td>
</tr>
<tr>
<td>LID (0005)</td>
<td>Type=Keyboard</td>
<td>DB=0438:07e4</td>
<td>FTT=0448:01e4</td>
</tr>
<tr>
<td>LID (0006)</td>
<td>Type=Printer</td>
<td>DB=0438:080c</td>
<td>FTT=0448:0238</td>
</tr>
<tr>
<td>LID (0007)</td>
<td>Type=Asynch</td>
<td>DB=0438:082c</td>
<td>FTT=0448:0280</td>
</tr>
<tr>
<td>LID (0008)</td>
<td>Type=SysTimer</td>
<td>DB=0438:084c</td>
<td>FTT=0448:02e8</td>
</tr>
<tr>
<td>LID (0009)</td>
<td>Type=RTCtimer</td>
<td>DB=0438:0860</td>
<td>FTT=0448:0328</td>
</tr>
<tr>
<td>LID (000a)</td>
<td>Type=SysService</td>
<td>DB=0438:087c</td>
<td>FTT=0448:0380</td>
</tr>
<tr>
<td>LID (000b)</td>
<td>Type=INTerrupt</td>
<td>DB=0438:08a0</td>
<td>FTT=0448:03cc</td>
</tr>
<tr>
<td>LID (000c)</td>
<td>Type=PointDevice</td>
<td>DB=0438:08d8</td>
<td>FTT=0448:0404</td>
</tr>
<tr>
<td>LID (000d)</td>
<td>Type=DMA</td>
<td>DB=0438:08f0</td>
<td>FTT=0448:044c</td>
</tr>
<tr>
<td>LID (000e)</td>
<td>Type=Security</td>
<td>DB=0438:0920</td>
<td>FTT=0448:04e4</td>
</tr>
<tr>
<td>LID (000f)</td>
<td>Type=POS</td>
<td>DB=0438:0938</td>
<td>FTT=0448:04f0</td>
</tr>
</tbody>
</table>
LID(0010)  Type=CMOSRam      DB=0438:0960  FTT=0448:0538
LID(0011)  Type=ErrorLog     DB=0438:0978  FTT=0448:0574
LID(0012)  Type=             DB=0438:0990  FTT=0448:05ac
LID(0013)  Type=Disk         DB=0438:09d8  FTT=0448:060c
LID(0014)  Type=anonymous    DB=0438:0a50  FTT=0448:0684
LID(0015)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(0016)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(0017)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(0018)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(0019)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(001a)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(001b)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(001c)  Type=Null         DB=0000:0000  FTT=0000:0000
LID(001d)  Type=Null         DB=0000:0000  FTT=0000:0000

Note: There is a formatting error that is illustrated in LID 12 and LID 14 lines. See description below of type= parameter for an explanation of this!

The fields displayed have the following meaning:

**LID**
Logical Device ID.
This is a sequential numbering of entries that appear in the table of LID entries. The entry, LID(0000), is however a dummy entry mapped by CDA:Type where the selector:offset of DB= are number of LID entries and offset to table of data pointers. Data pointer entries have one of the following forms:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off Len Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataPtr</td>
<td>+0 6</td>
</tr>
<tr>
<td>DLimit</td>
<td>+0 2 W</td>
</tr>
<tr>
<td>DOffset</td>
<td>+2 2 W</td>
</tr>
<tr>
<td>DSegment</td>
<td>+4 2 W</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off Len Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysPtr</td>
<td>+0 6</td>
</tr>
<tr>
<td></td>
<td>+0 2 W</td>
</tr>
<tr>
<td>PhysLSW</td>
<td>+2 2 W</td>
</tr>
<tr>
<td>PhysMSW</td>
<td>+4 2 W</td>
</tr>
</tbody>
</table>

**Type=**

This an interpretation of the device type (DevID) field taken from the corresponding Device Block. The following Type values may appear:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Used only for the LID(0000) dummy entry.</td>
</tr>
<tr>
<td>Null</td>
<td>signifies an unused entry (DB=0000:0000).</td>
</tr>
<tr>
<td>Internal</td>
<td>DevID=0000 used for internal BIOS calls.</td>
</tr>
<tr>
<td>Diskette</td>
<td>DevID=0001 Diskette device.</td>
</tr>
<tr>
<td>Disk</td>
<td>DevID=0002 Disk device.</td>
</tr>
<tr>
<td>Video</td>
<td>DevID=0003 Video device.</td>
</tr>
<tr>
<td>Keyboard</td>
<td>DevID=0004 Keyboard.</td>
</tr>
<tr>
<td>Printer</td>
<td>DevID=0005 Printer.</td>
</tr>
<tr>
<td>Asynch</td>
<td>DevID=0006 Asynchronous device.</td>
</tr>
<tr>
<td>SysTimer</td>
<td>DevID=0007 System Timer.</td>
</tr>
<tr>
<td>RTCTimer</td>
<td>DevID=0008 RTC Timer.</td>
</tr>
<tr>
<td>SysService</td>
<td>DevID=0009 SysService.</td>
</tr>
</tbody>
</table>
NMInterrupt - Devid=000a NMI Interrupt.
PointDevice - Devid=000b Pointer Device.
LightPen - Devid=000c Light Pen.
JoyStick - Devid=000d JoyStick.
CMOSRam - Devid=000e CMOS RAM.
DMA - Devid=000f DMA controller.
POS - Devid=0010 Programmable Option Select.
ErrorLog - Devid=0011 Error Log.
S/A Dump - Devid=0012 Stand Alone Dump.
Intersection - Devid=0013 I/O Port Allocation.
Audio tone - Devid=0014 Audio device.
Int/8259 - Devid=0015 Interrupt Controller.

Other Device Types are in use but are not translated to a predictable name.

For example:

Devid=0017 - SCSI Subsystem Interface
Devid=0018 - SCSI Peripheral

Where this occurs the Devid may be found at offset +8 of the device block.

DB=sel:off

sel:off address of the Device Block for the corresponding LID. The device block has the following standard header structure:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceBlock</td>
<td>+0</td>
<td>8</td>
<td></td>
<td>Device Block Header</td>
</tr>
<tr>
<td>DevBlength</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Device Block Length</td>
</tr>
<tr>
<td>Revision</td>
<td>+2</td>
<td>1</td>
<td>B</td>
<td>Revision</td>
</tr>
<tr>
<td></td>
<td>+3</td>
<td>1</td>
<td>B</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Logical ID</td>
</tr>
<tr>
<td>Devid</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Device ID</td>
</tr>
</tbody>
</table>

FTT=sel:off

sel:off address to the Function Transfer Table for this LID. The FTT has the following standard header structure:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTTTable</td>
<td>+0</td>
<td>16</td>
<td></td>
<td>Function Transfer Table Header</td>
</tr>
<tr>
<td>FStart</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Start Routine Entry Point</td>
</tr>
<tr>
<td>FInt</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Interrupt Routine Entry Point</td>
</tr>
<tr>
<td>FTimeO</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Start Routine Entry Point</td>
</tr>
<tr>
<td>FuncCount</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Count of Functions</td>
</tr>
<tr>
<td></td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

.D - Display an OS/2 System Structure
Display an OS/2 System Structure.

**Syntax:**

\[ .D \quad \text{SFT} \quad \text{addr} \]

- VPB
- DPB
- CDS
- KSEM
- DT
- DEV
- REQ
- MFT
- BUF
- BPB
- SEM32
- MUXQ
- OPENQ

**Parameters:**

*structure*

The structure type may take one of the following values:

- **SFT** Format a file system **System File Table** entry.
- **VPB** Format a file system **Volume Parameter Block**.
- **DPB** Format a file system **Drive Parameter Block**.
- **CDS** Format a file system **Current Directory Structure**.
- **KSEM** Format a **Kernel Semaphore**.
- **DT** Disk Trace in now obsolete.
- **DEV** Format a device driver header.
- **REQ** Format a device driver request packet.
- **MFT** Format a **Master File Table** entry.
- **BUF** Format a file system I/O buffer.
- **BPB** Format a **BIOS Parameter Block**.
- **SEM32** Format a 32-bit semaphore.
- **MUXQ** Format a **mutex** semaphore wait queue.
- **OPENQ** Format a 32-bit semaphore open queue.

*addr*

Specifies the address of the structure to be formatted. If omitted then the current **DS** selector value, offset 0 is assumed.

An address expression may be specified.

**Results & Notes:**

**Warning:**

`.D` will format OS/2 structures without any validation. It is entirely incumbent on the user to ensure that the address used does in fact point to the named structure. Failure to observe this caution will result in meaningless information being displayed.

The following are examples of each of the 13 formatted structures. Refer to the **System Reference** for a description of each formatted structure.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFT</td>
<td>System File Table Entry</td>
</tr>
<tr>
<td>VPB</td>
<td>Volume Parameter Block</td>
</tr>
<tr>
<td>DPB</td>
<td>Drive Parameter Block</td>
</tr>
<tr>
<td>CDS</td>
<td>Current Directory Structure</td>
</tr>
</tbody>
</table>
Note:

The Dump Formatter prior to fix pack 29 for Warp V3 did not format structures contained in segments less than 512 bytes, typically .D DEV and .D DPB. This is fixed from fix pack 29 of Warp V3 and base Warp V4.

System File Table Entry (SFT)

```plaintext
.d sft d0:8
  sf_ref_count: 0001
  sf_usercnt: 0000
  reserved: 00
  sf_flags(2): 0100:0000
  sf_chain: #0000:0000
  sf_MFT: fe7fb788
sfdFAT_firFILEclus: 5ad6
sfdFAT_cluspos: 09c8
sfdFAT_dirclus: 0000
sfdFAT_dirpos: 00
sfdFAT_name: sfdFAT_EAhandle: 0000
sfdFAT_plock: 0000
sf_NmPipeSfn: 0000
sf_codepage: 0000

```

System File Table Entry

Notes:

The sfdFAT_name is only meaningful for SFTs that represent open FAT file system files.

For a description of the SFT fields see the System File Table Entry (SFT) in the System Reference.

Volume Parameter Block (VPB)

```plaintext
## ln gdt_vpb
0138:00000098 os2krnl:DOSGDTDATA:GDT_VPB
##.d vpb 98:12
  vpb_flink: 0000
  vpb_blink: 008d
  vpb_ref_count: 007b
  vpb_cluster_mask: 41
  vpb_cluster_shift: 00
  vpb_first_FAT: 00b2
```
### Volume Parameter Block

**Notes:**

The selector for the VPB segment may be found by listing the symbol `GDT_VPB` and using its offset.

The handle of a VPB (hVPB) is the offset within the VPB segment.

The VPB segment has a unique owner ID, which may be determined using the `.M command. In the case of the VPB segment it is allocated from the kernel resident heap, so the true owner id is found in the heap header (or its extension - the trailer).

For a description of the VPB fields see the Volume Parameter Block (VPB) in the System Reference.

--------------------------------------------

### Drive Parameter Block (DPB)

```plaintext
.d dpb 4b8:c4
.dpb drive: 07
dpb_unit: 07
dpb_driver_addr: 0798:0000
dpb_next_dpb: 04b8:00e0
dpb_cbSector: 0200
dpb_first_FAT: 0001
dpb_toggle_time: 00000000
dpb_hVPB: 0012
dpb_media: f8
dpb_flags: 20
dpb_drive_lock: 0000
dpb_strategy2: 07a0:139c
```

##.m 04b8:0c4
Drive Parameter Block

Notes:

The DPB may be located from the VPB.

The DPB segment has a unique owner ID, which may be determined using the .M command. In the case of the VPB segment it is allocated from the kernel resident heap, so the true owner ID is found in the heap header (or its extension - the trailer).

For a description of the DPB fields see the Driver Parameter Block (DPB) in the System Reference.

Current Directory Structure (CDS)

>> locate the SAS file system section
##dw 70:12 11
0070:00000012 0074
##dw 70:74
0070:00000074 0fa4 fe70 00c0 07f8 00a8 0428 0000
0070:00000084 03c8 ffff ffff 0843 0000 0000 0000 0000
0070:00000094 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000a4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000b4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000c4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000d4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000e4 0000 0000 0000 0000 0000 0000 0000 0000

>> +8 into the file system section is the CDS RMP selector.
>> Can verify this by checking out the memory object owner.
##.m 828:0

>> Owned by the Kernel Resident Heap. Look at the header
##.m (%(828:0):10-18)
%fe7015bd4 000007d0 ff5c07b0 0000bd64 0000060d
%fe7015c4 049d0600 0aee0014 00000001 00000400

>> This is an attributed block so look at the trailer
##.m (%(828:0):10-4+60c 18)
%fe7015f0 c0000000 00000000 00000000 00000000 00000000
%fe7015f0 049d0600 0aee0014 00000001 00000400

##.mo ff61
ff61 cdsmrp
Now dump the CDS handle table for the process of interest

Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD  pPTDA  pTCB  Disp  SG  Name
0048#  0029  0004  0029  0001  blk  0200  ab805000  ab99b820  ab97fc20  1ed4  11  cmd

Except for driver 07 (H:) the current directory handle is null.
This implies that the current directory for drive H: is not the root. To see which it is, we need to locate the the CDS entry with handle 0x0090.

The RMP has a 0x14 byte header. Each entry is prefixed with a word length followed by the handle for that entry.
Starting with the first entry scan through until handle 0x0090 is located.

The CDS starts after the length prefix.

Current Directory Structure

Notes:
The selector for the CDS segment may be located from the SAS, as illustrated above, or form the storage at label CDSAddr.
The CDS RMP has a unique owner ID, which may be determined using the .M command. In the case of the CDS RMP, it is allocated from the kernel resident heap, so the true owner id is found in the heap header (or its extension - the trailer).
Kernel Semaphore (KSEM)

>> Intra-Process serialisation mutex KSEM imbedded in the PTDA.
>> This KSEM is sometimes referred to as "fscrit" (file system critical
>> section) since it is used to serialise file system activity within
>> a process.
.
.p#
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp GG Name
*000c# 0002 0000 0002 0004 blk 0804 ab78d000 ab997020 ab978420 1c9c 00 cntrl
##.d ksem %ab997020 -ptda_ptdasem-ptda_start
Signature : KSEM Nest: 0000
Type : MUTEX
Flags : 00
Owner : 0000 PendingWriters: 0000
##

>> MFT Shared KSEM imbedded at the beginning of the MFT
##
#.d sft d0:8
 sf_ref_count: 0001 sf_mode: 00a0
 sf_usercnt: 0000 sf_hVPB: 0012
 reserved: 00 sf_cftime: 0000
 sf_flags(2): 0000:0000 sf_atime: 0000
 sf_FSC: #00c8:0000 sf_mtime: 0000
 sf_chain: #0000:0000 sf_size: 000bb135
 sf_MFT: fe7fb788 sf_position: 0008d90
##
#.d ksem %fe7fb788
Signature : KSEM Nest: 0000
Type : SHARE
Flags : 00
Owner : 0000 PendingWriters: 0000
##

>> Slot 49 is blocked. So we proceed by finding out what the BlockId
>> represents by finding its owner.
##
#.pb 49
Slot Sta BlockID Name Type Addr Symbol
0049 blk fe83bdf4 warp_d
##
#.m %0fe83bdf4
*har par cpg va flg next prev link hash hob hal
072c %feaf8dd2 00000040 149 072d 072b 0003 0000 0003 0025 hptda=0878
 hal=0025 pal=%fe83bdf4 hal=072c hptda=0878 pgoff=00000 f=021
 har par cpg va flg next prev link hash hob hal
0003 %feaf04c 00000040 001 0002 0023 0000 0000 0003 0000 =0000
 har hobnxt flgs own hme sown, cnt ln at xf
0003 072c fec5 0000 f1ec 0000 00 01 00 00 vmkrhw
 pvmli cs eip phblock cpg va flg hptda hob sig csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f

>> Block Id is in the kernel resident heap - assume that it is at the
>> beginning of a data portion of a heap block.
>> Dump the header.
##
#.dd %0fe83bdf4-10 18
%fe83bde4 00000000 bd100000 fe83fe83 ff7e0018
Kernel Semaphore

Notes:

KSEMs are usually found imbedded in system control blocks for serialisation and sharing purposes.

- Dynamically allocated KSEMs are allocated out of one of the kernel heaps.
- Virtual Device Driver semaphore helper services result in KSEMs.
- Under the ALLSTRICT kernel only, the KSEM has a signature field. This is manufactured by the .D command for non-ALLSTRICT kernels. Under the ALLSTRICT kernel the presence of a KSEM may be verified by dumping the KSEM in bytes. Offset +0x0 is where the signature is located.
- The owner field refers to the slot number of the semaphore owner.

When a thread blocks on a KSEM the following addresses are used as the BlockId:

- MUTEX KSEM: Address of the beginning of the KSEM structure.
- SHARED KSEM: Address of the Pending Readers count field within the KSEM structure.
- EXCLUSIVE KSEM: Address of the Pending Writers count field within the KSEM structure.

To format KSEM from a BlockId, locate the beginning of the KSEM, either by dumping a few bytes before the BlockId address. The signature will be visible under the ALLSTRICT kernel, which is at the beginning of the KSEM. If the KSEM is allocated from one of the Kernel's heaps then the KSEM object id, 0xff7e will occur in the heap header which prefixes the beginning of the KSEM.

For a description of the KSEM structure see the Kernel Semaphore Structure in the System Reference.

--------------------------------------------

Physical Device Driver Header (DEV)

>> Driver header address taken from the VBP with handle 12:

.d vpb 98:12

vpb_flink: 0000  vpdpFAT_cluster_mask: 41
vpb_blink: 008d  vpdpFAT_cluster_shift: 00
vpb_ref_count: 007a  vpdpFAT_first_FAT: 00b2
vpb_search_count: 0000  vpdpFAT_MAX_COUNT: a8
vpb_first_access: 00  vpdpFAT_root_entries: 0004
vpb_signature: 444a  vpdpFAT_first_sector: 885c0400
vpb_flags(2): 02:00  vpdpFAT_MAX_CLUSTER: 410e
vpb_PSC: #00c8:0008  vpdpFAT_DIR_SECTOR: aa04a800
vpi_ID: 26715015  vpdpFAT_MAX_CLUSTER: 410e
vpi_pDPB: #04b8:00c4  vpdpFAT_MEDIA: 0d
vpi_cbSector: 0200  vpdpFAT_FREE_CNT: 04a8
vpi_totsec: 0007cfe0  vpdpFAT_FREE_ENTRYSIZE: b2
vpi_trksec: 0020  vpdpFAT_free_cnt: 04a8
vpi_nhead: 0040  vpdpFAT_FREE_ENTRYSIZE: b2
vpi_pDCS: #0000:0000  vpdpFAT_IDsector: 00000000
vpi_pVCS: #0000:0000  vpdpFAT_ACCESS: 0000
vpi_pVCS: #0000:0000  vpdpFAT_ACCESS: 0000

--------------------------------------------
Physical Device Driver Header

Notes:

The Device Header appears in one of two formats, depending on whether the device supports multiple units or not.

DevInt is not the interrupt routine offset, as it was for DOS device drivers. Under OS/2 this is the offset to the Inter-Device Driver Communications (IDC) Entry Point.

For a description of the DEV fields see the Physical Device Driver Header (DEV) in the System Reference.

Device Driver (Strategy 1) Request Packet (REQ)

>> The two request packet pools for general device driver use:
Device Driver Request Packets

Notes:

Request Packets are allocated from one of three pools:

Strategy 1 request pool
Strategy 2 request pool
Swapper request pool

Each thread is pre-assigned a strategy 1 request packet. If this is in use when a device driver tries to allocate another then a packet is allocated from the strategy 2 pool for strategy 1 use.

Asynchronous Read and Write requests are implemented in DOSCALL1.DLL by creating multiple threads on which to run the parallel I/O requests.

.D REQ does not format Strategy 2 format Request Packets.
For a description of the Request Packet fields see the Device Driver Request Packer in the System Reference.

--------------------------------------------

Master File Table Entry (MFT)

>> Display the SFT for SFN 20

##.d sft d0:(83*20+8)

sf_ref_count: 0001  sfi_mode: 00c2
sf_usercnt: 0000  sfi_hVPB: 0000
reserved: 00  sfi_ctime: 0000
sf_flags(2): 00c0:0000  sfi_cdate: 0000
sf_devptr: #0af0:0000  sfi_atime: 0000
sf_FSC: #00c8:ff40  sfi_adate: 0000
sf_chain: #00d0:170f  sfi_mtime: b19d
sf_MFT: fe82ff7c  sfi_mdate: 1f5f
sfdFAT_firFILEclus: 0000  sfi_size: 00000000
sfdFAT_cluspos: 0000  sfi_position: 00000000
sfdFAT_l

>> From the SFT display the MFT

,.d mft %fe82ff7c

mft_ksem:
Signature : KSEM  Nest: 0000
Type : SHARE  Readers: 0000
Flags : 01  PendingReaders: 0000
Owner : 0000  PendingWriters: 0000
mft_lptr: 0000  mft_sptr: 00d0:08bb
mft_lptr: 0000  mft_sptr: 00d0:08bb
mft_pMap: 00000000  mft_sMap: 013e  mft_signature: 466d
mft_CMapKSem:
  mft_hvpb: 0000  mft_opflags: 0000  mft_flags: 0000
  mft_sptr: 00d0:2045
mft_name: \DEV\MOUSE$

>> Display the SFT for SFN 40

##.d sft d0:(83*40+8)

sf_ref_count: 0001  sfi_mode: 1302
sf_usercnt: 0000  sfi_hVPB: 008d
reserved: 00  sfi_ctime: 0000
sf_flags(2): 0000:0000  sfi_cdate: 0000
sf_devptr: #0000:0000  sfi_atime: 0000
sf_FSC: #00c8:0008  sfi_adate: 0000
sf_chain: #00d0:214b  sfi_mtime: 0000
sf_MFT: fe6f190c  sfi_mdate: 0000
sfdFAT_firFILEclus: 470c  sfi_size: 00000000
sfdFAT_cluspos: 09c8  sfi_position: 00000000
sfdFAT_l

>> From the SFT display the MFT

,.d mft %fe6f190c

mft_ksem:
Signature : KSEM  Nest: 0000
Type : SHARE  Readers: 0000
Flags : 01  PendingReaders: 0000
Owner : 0000  PendingWriters: 0000
mft_lptr: 0000  mft_sptr: 00d0:2045
mft_lptr: 0000  mft_sptr: 00d0:2045
mft_pMap: 00000000  mft_sMap: 00a3  mft_signature: 466d
mft_CMapKSem:
  mft_hvpb: 008d  mft_opflags: 0000  mft_flags: 0001
  mft_sptr: 00d0:2045
mft_name: D:\SWAPPER.DAT

Master File Table Entries
Notes:

The MFT is entry may be located from each SFT that represents an open instance of a file.
The MFT points to the most recent SFT open instance of the file.
The Dump Formatter prior to fix pack 29 for Warp V3 did not format the MFT correctly under the ALLSTRICT kernel. This is fixed from fix pack 29 of Warp V3 and base Warp V4.

For a description of the MFT field, see the Master File Table Entry (SFT) in the System Reference.

--------------------------------------------

File System Buffer (BUF)

>> Locate the file system buffer segment

```c
#include <gdt_buffers>
0138:000000a8 os2krnl:DOSGDTDATA:GDT_Buffers

#include

buf_next: 001c            buf_prev: ffff           buf_freeLink: 0000
buf_flags:  02              buf_hVPB: 0279           buf_sector: 00000001
buf_tid: 00              buf_wrtcnt: 02           buf_wrtcntinc: 0096
buf_fill: 0000

buf_next: 044c            buf_prev: 0234           buf_freeLink: 0000
buf_flags:  02              buf_hVPB: 0279           buf_sector: 00000001c
buf_tid: 00              buf_wrtcnt: 02           buf_wrtcntinc: 0096
buf_fill: 0000

buf_next: 0664            buf_prev: 044c           buf_freeLink: 0000
buf_flags:  02              buf_hVPB: 0279           buf_sector: 00000001c
buf_tid: 00              buf_wrtcnt: 02           buf_wrtcntinc: 0096
buf_fill: 0000

buf_next: 087c            buf_prev: 001c           buf_freeLink: 0000
buf_flags:  02              buf_hVPB: 0279           buf_sector: 00000001c
buf_tid: 00              buf_wrtcnt: 02           buf_wrtcntinc: 0096
buf_fill: 0000

>> Find the volume these buffers are assigned to.

#include
gdt_vpb
0138:00000098 os2krnl:DOSGDTDATA:GDT_VPB

#include

vpb_flink: 01fe            vpb_freeLink: 00000000
vpb_blink: 02f4            vpb_registro牒: 00000001
vpb_ref_count: 004e         vpb_registro牒: 00000001c
vpb_search_count: 0000       vpb_registro牒: 00000001c
vpb_first_access: 09        vpb_registro牒: 00000001c
vpb_signature: 444a         vpb_registro牒: 00000001c
vpb_flags[2]: 02:40         vpb_registro牒: 00000001c
vpb_SPC: #0000:ff40         vpb_registro牒: 00000001c
vpi_ID: e2ea4414            vpb_registro牒: 00000001c
vpi_pDPB: #04b8:0038        vpb_registro牒: 00000001c
vpi_cbSector: 0200          vpb_registro牒: 00000001c
vpb_FSC: #0000:ff40        vpb_registro牒: 00000001c
vpb_FIC: #0000:ff40        vpb_registro牒: 00000001c
vpb_FIC: #0000:ff40        vpb_registro牒: 00000001c
```
>> The file system buffer segment is assigned a unique object owner
>> id.

File System Buffer (BUF)

Notes:

File system buffers are allocated out of a buffer segment whose selector may be located either from the SAS File System section, offset +0xa or from symbol GDT_BUFFERS.

The buffer segment contains a header of length +0x1c.

Header Offset +0x0 gives the offset to the head of the list of most recently used buffers.

Header Offset +0x4 gives the offset to the tail of the list of most recently used buffers.

Each buffer contains a 0x18 byte header followed by 0x200 bytes of data. The buffer header is what is formatted by .D BUF.

For a description of the Buffer Header fields, see the File System Buffers in the System Reference.

BIOS Parameter Block (BPB)
SectorsPerCluster: 01
ReservedSectors: 0001
NumberOfFATs: 00
RootEntries: 0010
TotalSectors: 0000
MediaDescriptor: f0
SectorsPerFAT: 0001
SectorsPerTrack: 0009
Heads: 0001
HiddenSectors: 00000000
BigTotalSectors: 003fffff

BIOS Parameter Block (BPB)

Notes:

Two system BPBs are locatable at symbols BootBPB and minimumBPB.
Others are pointed to from the Device Driver Request Packet for DosDevIOCtl command code 2 (build BPB).
See .D REQ Command for information on formatting Device Driver Request Packets.

For a description of the BPB fields, see the BIOS Parameter Block in the System Reference.

32-Bit Semaphore Structures (SEM32, OPENQ and MUXQ)

##.pb 25
Slot Sta BlockID Name Type Addr Symbol
0025 blk fe81d2d0 pmshell Sem32 8001 004b hevLazyWrite

##.d sem32 %fe81d2d0
Type: Shared Event
Flags: Reset
pMUXQ: 00000000
Post Count: 0000
pOpenQ: fe56eb10
pName: fd074e98
Create Addr: 13f60088

##.pb 2f
Slot Sta BlockID Name Type Addr Symbol
002f blk fe86ffdc pmshell

##.d sem32 %fe86ffdc
Type: Shared Event
Flags: Reset
pMUXQ: 00000000
Post Count: 0000
pOpenQ: fe56eb02
pName: NULL (anonymous)
Create Addr: 12d16b48

##.pb 30
Slot Sta BlockID Name Type Addr Symbol
0030 blk fe86fe58 pmshell Sem32 0001 00ce hevSleeper

##.d sem32 %fe86fe58
Type: Private Event
Flags: Reset
pMUXQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 13f62f28
Three types of Event Semaphore

Notes:

32-bit semaphores may be Event or Mutex in type, private or shared in scope and if shared, named or anonymous.

The BlockId of a thread waiting on a 32-bit semaphore is the address of the semaphore structure. The **Type** field in the `.PB command usually indicates a 32-bit semaphore when in use, however this is not always the case. The next example shows how to determine precisely whether the blocidk points to a 32-bit semaphore.

```
##.pb 33
Slot Sta BlockID Name Type Addr Symbol
0033 blk fe86falc pmshell
##.m %0fe86falc
*har par cpg va flg next prev link hash hob hal
0003 %fe8ef04c 00000400 %feef000 001 0002 023 0000 0003 0000 =0000
hob har hobnat flgs own hmtc sown, cnt lt st xf
0003 0003 fec5 0000 ffecc 0000 0000 00 01 00 00 vmxhrw
pvmli cs eip phlock cpg va flg hptda hob sig csig
%fe8e380 002d 0a6800a5 8ee24c3 0000 %fe8e0c00 0005 024b 0003 ea9f ea9f
##dd %0fe86fa1c-10 18
%fe86fa0c 12d15b4c 54564553 ab97d220 ffc20018
%fe86falc 00000010 00000000 c4280001 455000a7
##.mo ffc2
ff2 semstruc
##.d sem32 %0fe86falc
Type: Private Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 00a7c428
##
```

How to determine whether a BlockId points to a 32-bit semaphore.

Notes:

Except for RAMSEM, MUXWAIT, ChildWait and private conventions the BlockID is an address of a structure or routine that relates to the resource or event being waited for.

The **.M command** is used to identify the owner of the BlockId. In this case it is the kernel resident heap.

Each resident heap block is prefixed with a 4 byte header. If the low order bit is 0 then the high word of the header contains the owner of the heap block.

32-bit Semaphore structures are allocated from regular resident heap blocks. Thus the owner id may be seen by displaying storage before the BlockId address.

```
##.pb 56
0056 blk fe88ad8c mutxwait Sem32 8001 0090 _WINOS2_Settings + 77
##.d sem32 %fe88ad8c
Type: Shared Mutex
Flags:
pMuxQ: fe88ab94
Request Ct: 0001
Owner: 0055
Requester Ct: 0001
pOpenQ: fe5724c2
pName: fd084368
Create Addr: 00022e98
##.d openg fe5724c2
```

PID Open Count
### Mux Wait Semaphores

**Notes:**

- **pOpenQ** points to an Open Queue Structure, that list all processes that have access to the 32-bit semaphore. This is formatted using `.D OPENQ.`
- **pName** points to the semaphore name, when not anonymous.
- **pMuxQ** points to a MUXQ structure, that lists any 32-bit MUX wait semaphore address lists that have included this semaphore. In this example we see one MUX list.

The MUX list may be formatted using `.D SEM32.`

Instead of a **pMuxQ**, the MUX semaphore contains a pointer to the semaphore record (`SR Pointer`) and a count of the number of semaphores in the list (`SR Count`).

There is no special formatting command for the `SR Structure` - it has to be view by displaying storage directly. In this case we see then length, flags and three semaphore handles each followed by the user correlator.

For a description of the 32-bit Semaphore Structures, see the 32-bit Semaphore Structures in the System Reference.

---

.I - Swap in Storage

---
Page in a TSD or a Page of Virtual Storage from the Swapper file.

Syntax:

```
.I                                      addr               `.B
.D
.T                                              `B                slot
.D
```

Parameters:

**T**

When specified it requests the Kernel Debugger page in the TSD for a specified thread slot.

One TSD is assigned to each thread slot. If the registers for an out-of-context thread need to be examined then it may be necessary to swap in the TSD for that slot, since the ring 3 stack frame is stored in the TSD when a thread enters the Kernel. The presence or absence of the TSD for a given slot may be deduced by the presence or absence of a value for the **Disp** field of the **.P** command.

If the **T** option is omitted it requests the Kernel Debugger page in the page of virtual storage that encompasses the specified **addr**.

**B**

When specified requests that all breakpoints be re-instated, including those which the current CS:EIP may be addressing.

This parameter is effectively obsolete since breakpoints at the current CS:EIP are correctly handled by the Breakpoint Commands.

**D**

Specifies that a page-in request be scheduled for the Kernel Debugger Daemon thread to execute. In most cases a page-in operation may be performed synchronously, but under the following conditions it is prohibited:

- When an interrupt is being handled (that is, not at task time)
- When a swapping operation is pending (TCBswapping (TCB + 0x1a1) not 0)
- When a the current thread is blocked (TK_WF_SLEEPING (0x40) is set in TCBWakeFlags (TCB + 0x162))
- When in ring0 and InDos is 0

When one of these conditions occurs the page-in request may be scheduled for execution asynchronously by the Debugger Daemon thread by use of the **D** parameter. If the request is successfully scheduled the user is invited to enter the **G** command. The system will dispatch the Daemon thread, in time, which will attempt the page-in request. The Daemon returns control to the debug console using an **INT 3** interrupt.

**addr**

This specifies the virtual address of the page to be paged in. The address is effectively rounded down to the nearest 4K page boundary.

**Note:** A selector:offset address specification can only be used if the selector does not reference the packed area of the LDT. If it does then a linear address must be supplied by the user.

**slot**

Specifies the thread slot number of the TSD to be paged in. The default slot is the current slot of the debugger's default slot if overridden with the **.S** command.

Results & Notes:

When an asynchronous page-in is requested the Kernel Debugger will prompt the user with one of the following:

```
 task|addr %nnnn|%nnnnnnnn, LDT entry address %nnnnnnnn queued, G to continue
```
task|addr %nnn|%nnnnnn queued, G to continue

TSD for slot s queued, G to continue

depending upon combination of parameters specified.

On successful completion of a synchronous page-in the user will be prompted with the command prompt.

If .I is unable to complete the request the OS/2 system error code will be displayed (in decimal) in the following message:

OS/2 error code nt

refer to the Control Programming reference or to bseerr.h C header file for an interpretation of the error code.

--------------------------------------------

.H - Display Dump File Header Information

Display dump file header information saved by the stand alone dump program in the first sector (512 bytes) of the dump file.

Syntax:

` .H `

Parameters:

None.

Results & Notes:

This command displays the following information:

`.h
Dump File Header Info:
  Start Addr1: 0
  End Addr1: 2623213
  Total Disks: 9
  Flag: 11
  Ending addresses by disk:
    2623213   6634079   9846551   12950323
    14965147  17345751  19711393
    22092095  25165823
#`  

Each of the fields displayed has the following meaning:

Start Addr1: The lowest physical address dumped.

End Addr1: The highest physical address dumped.

Total Disks: The number of disk volumes the dump data set spans. When the dump is taken to hard disk then the number of volumes is one.

Flag: Indicates whether the dump file required decompressing. 0 indicates a compressed dump and 11 a decompressed
dump.

Ending Addresses by disk
   Shows the range of physical memory dumped to each disk volume.

---------------------------------------------

.I (DF) - Show Dump State

Display the dump state.

Syntax:

   .I

Parameters:

None.

Results & Notes:

This command displays the following summary information:

------------------------------------------------------------------
#.i
PROCESS slot:1c Pid:0003 Ord:0013
PTDA    handle=0088 address=%7bcd5844
MTE     handle=018a address=%fdefdf78 (PMSHL32)
SMTE    address=%fc9a4c48
LDT     handle=0187 address=%7a597000
 CODE:  user (cs:eip)#005b:17d679f2 cbargs=
 STACKS: user (ss:esp)#0053:01382cbc(active)
          ring2(ss:esp)#09be:00004000(bottom)
          ring0 tcbframe=%7bbd4f54 bottom=%7bbd4f9c
------------------------------------------------------------------

Each of the fields display has the following meaning:

slot: The current thread slot at the time the dump was taken. This value is taken from the TaskNumber global variable.

Pid: The current pid when the dump was taken. This value is taken from the Pid global variable.

Ord: The Tid of the current thread at the time the dump was taken. This value is taken from the TCBOrdinal (TCB+ 0x0) of the current TCB.

handle= The VMOB handle that represents the control block named to the left.

.I displays this information for the PTDA, MTE, SMTE and LDT associated with the current thread when the dump was taken.

See the .MO command form more information.

address= The address of the object whose name and handle are given on the same line of display.

user (cs:eip) The current user CS:EIP when the dump was taken. See the .R command for related information.

cbargs=
The call gate argument count if the current task has made a privilege level transition. See the .PU command for further information.

user (ss:esp)      The current user SS:ESP when the dump was taken. See the .R command

ring2(ss:esp)      The current ring 2 SS:ESP as saved in TCBCpl2_SS (TCB + 0x1bc) and TCBCpl2_ESP (TCB + 0x1b8) fields of the current TCB.

ring0 tcbframe=    The current (or last) kernel entry stack frame pointed to be TCB_pFrameBase (TCB + 0x3c) when the current thread made a call or transition to the kernel.

bottom=           The base of the ring 0 stack (in its all contexts addressable form) for the current thread.

.K - Display User Stack Trace

Display the user stack-trace for a given thread slot.

Syntax:

```
.K
.KS      #
.KB      *

slot
```

Parameters:

.K       Display stack frame trace assuming the default operation size from the descriptor associated with the code selector of the user registers for the specified slot.

.KS      Display frame trace assuming an operation size of 16-bits (small-model).

.KB      Display frame trace assuming an operation size of 32-bits (big-model).

slot     Display stack trace for thread slot slot.

The following short-hand may be used for the slot number:

*       The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

    __TaskNumber

#       The debugger default thread slot. This defaults to the current slot unless overridden by the .S command.

If no slot number is given then all thread slots are displayed, grouped by process.

Results & Notes:

The .K command operates as a K command but with the starting stack frame and code segment address implicitly determined from the user's register as displayed by the .R command.
The output from the .K command displays exactly as the K command but with the slot-number prefixed to the return address when an out-of-context stack trace is displayed. See example output below.

Warning:
The .K command is subject to the same limitations as noted for the K command. See the K command description for details.

Example output from an out-of-context stack trace:

```
-------------------------------------------------------------------------------
##.S 8
##.K 37
0037|a6e7:0000006f 03d4 0000 00c5 006f
0037|a6e7:00000000 0000 0000 0000 0000
##-------------------------------------------------------------------------------
```

--------------------------------------------

.LM - Format Loader Structures (MTE, SMTE, OTE and STE)

Display selected information from the MTE and SMTE of one or more loaded modules. Optionally format the associated STE or OTE.

Syntax:

```
.LM
```

Parameters:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Format information about each object of each load module. For 32-bit modules selected fields from the Object Table Entry (OTE) are displayed. For 16-bit modules selected fields from the Segment Table Entry (STE) are displayed.</td>
</tr>
<tr>
<td>I</td>
<td>Select Installable File System Driver modules only.</td>
</tr>
<tr>
<td>L</td>
<td>Select Dynamic Link Library modules only. (This includes DLLs and any other modules which are not specifically selectable by the other options.)</td>
</tr>
<tr>
<td>P</td>
<td>Select Physical Device Drivers modules only.</td>
</tr>
<tr>
<td>V</td>
<td>Select Virtual Device Drivers modules only.</td>
</tr>
<tr>
<td>X</td>
<td>Select Executable modules (.EXE) only.</td>
</tr>
</tbody>
</table>

hmte

 specifies the handle of the memory object assigned to the MTE structure to be formatted.

addr

 specifies the address of the MTE to be formatted.

name

 specifies the name (excluding the file extension and path). The MTE matching this name will be formatted. The name
must be specified as a quoted string.

This option requires the SMTE to be present in storage. See below for information on how to make the SMTE present.

The default specification is to scan the entire MTE chain without formatting corresponding STEs or OTEs.

**Results & Notes:**

From fix pack 29 for Warp V3 and base Warp V4 the following changes have been made:
- The LM has been fixed for the dump formatter so that it displays the short name of modules when the SMTE is swapped out.
- The I parameter has been introduced.
- With some previous version of the Dump Formatter the LMP and LMV commands did not always display output.

From OS/2 Warp V3.0 fix pack 40, OS/2 Warp V4.0 fix pack 10 and OS/2 Warp E-Server the swappable structures referenced by .LM can be forced to be allocated from resident memory by using the OTE option of the RASKDATA CONFIG.SYS statement. This will avoid the possibility that .LM might not be able to display information some modules.

The MTE chain is scanned from global symbol:

```
__mte_h
```

When OTE/STE formatting is not requested output appears as follows:

```
.lm
hmte=0293 pmte=%f4f1a38 mflags=06903140 e:\os2\tools\mrfile32.exe
hmte=027f pmte=%f4f1c80 mflags=06903142 !pulse
hmte=0272 pmte=%f4f1db4 mflags=06903152 c:\os2\cmd.exe
hmte=00a8 pmte=%f8e77a8 mflags=0698b194 c:\os2\dll\display.dll
hmte=017a pmte=%f8e15abc mflags=0698b198 c:\os2\dll\bvhwndw.dll
hmte=010e pmte=%f8e282dc mflags=0691b180 ???
hmte=0101 pmte=%f8e166b8 mflags=0691b180 ???
hmte=00f9 pmte=%f8e16cd4 mflags=0691b180 c:\os2\mdos\vmfreefile32.exe
hmte=00f5 pmte=%f8e16de0 mflags=0691b180 c:\os2\mdos\vbios.sys
hmte=0072 pmte=%f8f2c919 mflags=0692b180 mvdma.dll
hmte=0006 pmte=%f8f2d60 mflags=0692f880 doscalls.dll
hmte=01c8 pmte=%f8f5d78 mflags=0698b1c8 c:\os2\dll\times.fon
hmte=01c6 pmte=%f8f7718 mflags=0698b1c8 c:\os2\dll\helv.fon
hmte=00d5 pmte=%f8f32e60 mflags=0608f1c9 c:\os2\pmdd.sys
hmte=00d6 pmte=%f8f32f4 mflags=0608f1c9 c:\os2\dos.sys
hmte=00cd pmte=%f8f4f64 mflags=0608f1c9 c:\os2\testcfg.sys
hmte=00cc pmte=%f8f4fb40 mflags=0628a1c9 c:\os2\hpfs.ifs
hmte=00a2 pmte=%f8f4fb40 mflags=0408e1c9 c:\os2\dasd.dmd
hmte=00a1 pmte=%f8f3288c mflags=0408e1c9 c:\ibm2scsi.add
hmte=009f pmte=%f8f2f18 mflags=0408e1c9 c:\ibm2fpw.py.add
hmte=0096 pmte=%f8f4f60 mflags=0408e1c9 c:\print02.sys
hmte=0093 pmte=%f8f2efb8 mflags=0408e1c9 c:\clock02.sys
#
```

The fields formatted have the following meaning:

- **hmte**: Handle of the memory object occupied by this MTE. Taken from `mte_handle`
- **pmte**: Linear address of this MTE
- **mflags**: Flag field 1 taken from `mte_flags1`. These flags have the following interpretation:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOAUTODS</td>
<td>0x00000000</td>
<td>No Auto DS exists</td>
</tr>
<tr>
<td>SOLO</td>
<td>0x00000001</td>
<td>Auto DS is shared</td>
</tr>
<tr>
<td>INSTANCEDS</td>
<td>0x00000002</td>
<td>Auto DS is not shared</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INSTLIBINIT</td>
<td>0x00000004</td>
<td>Per-instance Libinit</td>
</tr>
<tr>
<td>GINISETUP</td>
<td>0x00000008</td>
<td>Global Init has been setup</td>
</tr>
<tr>
<td>NOINTERNFIXUPS</td>
<td>0x00000010</td>
<td>internal fixups in .EXE-.DLL applied</td>
</tr>
<tr>
<td>NOEXTERNFIXUPS</td>
<td>0x00000020</td>
<td>external fixups in .EXE-.DLL applied</td>
</tr>
<tr>
<td>CLASS_PROGRAM</td>
<td>0x00000040</td>
<td>Program class</td>
</tr>
<tr>
<td>CLASS_GLOBAL</td>
<td>0x00000080</td>
<td>Global class</td>
</tr>
<tr>
<td>CLASS_SPECIFIC</td>
<td>0x000000C0</td>
<td>Specific class, as against global</td>
</tr>
<tr>
<td>CLASS_ALL</td>
<td>0x00000000</td>
<td>nonspecific class - all modules</td>
</tr>
<tr>
<td>CLASS_MASK</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>MTEPROCESSED</td>
<td>0x00000100</td>
<td>MTE being loaded</td>
</tr>
<tr>
<td>USED</td>
<td>0x00000200</td>
<td>MTE is referenced</td>
</tr>
<tr>
<td>DOSLIB</td>
<td>0x00000400</td>
<td>set if DOSCALL1</td>
</tr>
<tr>
<td>DOSMOD</td>
<td>0x00000800</td>
<td>set if DOSCALLS</td>
</tr>
<tr>
<td>MTE_MEDIAFIXED</td>
<td>0x00001000</td>
<td>File Media permits discarding</td>
</tr>
<tr>
<td>LDRINVALID</td>
<td>0x00002000</td>
<td>module not loadable</td>
</tr>
<tr>
<td>PROGRAMMOD</td>
<td>0x00000000</td>
<td>program module</td>
</tr>
<tr>
<td>DEVDRVMOD</td>
<td>0x00004000</td>
<td>device driver module</td>
</tr>
<tr>
<td>LIBRARYMOD</td>
<td>0x00008000</td>
<td>DLL module</td>
</tr>
<tr>
<td>VDDMOD</td>
<td>0x00010000</td>
<td>VDD module</td>
</tr>
<tr>
<td>MVDMMOD</td>
<td>0x00020000</td>
<td>Set if VDD Helper MTE (MVDM.DLL)</td>
</tr>
<tr>
<td>INGRAPH</td>
<td>0x00040000</td>
<td>In Module Graph</td>
</tr>
<tr>
<td>GINIDONE</td>
<td>0x00080000</td>
<td>Global Init has finished</td>
</tr>
<tr>
<td>MTEADDRALLOCED</td>
<td>0x00100000</td>
<td>Allocate specific or not</td>
</tr>
<tr>
<td>FSDMOD</td>
<td>0x00200000</td>
<td>FSD MTE</td>
</tr>
<tr>
<td>FSHMOD</td>
<td>0x00400000</td>
<td>FS helper MTE</td>
</tr>
<tr>
<td>MTELONGNAMES</td>
<td>0x00800000</td>
<td>Module supports long-names</td>
</tr>
<tr>
<td>MTE_MEDIACONTIG</td>
<td>0x01000000</td>
<td>File Media contiguous memory req</td>
</tr>
<tr>
<td>MTE.MEDIA16M</td>
<td>0x02000000</td>
<td>File Media requires mem below 16M</td>
</tr>
<tr>
<td>MTEIOPALLOWED</td>
<td>0x04000000</td>
<td>Module has IOPL privilege</td>
</tr>
<tr>
<td>MTEPORTHOLE</td>
<td>0x08000000</td>
<td>porthole module</td>
</tr>
<tr>
<td>MTEMODPROT</td>
<td>0x10000000</td>
<td>Module has shared memory protected</td>
</tr>
<tr>
<td>MTENEWTERM</td>
<td>0x20000000</td>
<td>Newly added module</td>
</tr>
<tr>
<td>MTEDLLTERM</td>
<td>0x40000000</td>
<td>Gets instance termination</td>
</tr>
<tr>
<td>MTESYMLOADED</td>
<td>0x80000000</td>
<td>Set if debugger symbols loaded</td>
</tr>
</tbody>
</table>

The full path name for the module is displayed to the right of the mflags field. The name is taken from the smte_path of the SMTE. If the SMTE is swapped out then the the name is taken from mte_modname (the .DEF file link edit name) and prefixed with an ! symbol.

Where no path information is given then the module is predefined by the system and does not exist separately as a load module file.
The STE and OTE are displayed when the O option is specified. These tables are accessed from the address at SMTE+0x1c. This requires that the SMTE be present in storage. If it is not then the following is returned:

```
???
```

swappable MTE - swapped

To page in the SMTE use `.LM` without parameters to obtain the MTE address from the `pmte` field. The SMTE address is at MTE + 0x4. Use the `.I` command to page in the SMTE storage.

Under the Dump Formatter nothing can be done, however use of the OTE option of the RASKDATA CONFIG.SYS statement will guarantee that structures used by `.LM` are retained in resident memory.

For a 16-bit module the STE is formatted as follows:

```
# .lmo 'hpfs'
```

```
htme=00cc pmte=%fdff4fb40 mflags=0628a1c9 c:\os2\hpfs.ifs
seg sect psiz vsiz hob sel flags
0001 0003 eb24 eb24 0000 0668 8d60 code shr prel rel
0002 0079 d22f d230 0000 0670 8d60 code shr prel rel
0003 00e3 07b5 07b8 0000 0678 8d60 code shr prel rel
0004 00e8 0d8a 0d8c 0000 0680 8d60 code shr prel rel
0005 00f0 0d6e 19c2 0000 0688 8d41 data prel rel
0006 00f7 03fb 03fc 0000 0690 8c41 data prel
0007 00f9 0084 0084 0000 0698 8d41 data prel rel
0008 00fa 0010 0014 0000 06a0 8d41 data prel rel
0009 00fb 0238 0238 0000 06a8 8d41 data prel rel
```

The STE fields formatted have the following meaning:

- **seg**: Segment number. This is a sequential index of module segments. Index entries appearing in the link-edit map will correspond with these values.

- **sect** *(ste_offset)* Offset in file to segment data.

- **psiz** *(ste_size)* File data size

- **vsiz** *(ste_minsiz)* Minimum allocation size

- **hob** *(ste_seghdl)* Memory object handle of segment data.

- **sel** *(ste_selector)* Selector assigned to this segment.

- **flags** *(ste_flags)* Segment type and attribute flags. These interpretations of these are displayed to the right of the flag word. They are assigned as follows:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>.imo</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STE_CODE</td>
<td>0x0000</td>
<td>code</td>
<td>code segment type</td>
</tr>
<tr>
<td>STE_DATA</td>
<td>0x0001</td>
<td>data</td>
<td>data segment type</td>
</tr>
<tr>
<td>STE_PACKED</td>
<td>0x0002</td>
<td></td>
<td>segment is packed</td>
</tr>
<tr>
<td>STE_SEMAPHORE</td>
<td>0x0004</td>
<td></td>
<td>segment semaphore</td>
</tr>
<tr>
<td>STE_ITERATED</td>
<td>0x0008</td>
<td>iter</td>
<td>segment data is iterated</td>
</tr>
</tbody>
</table>
For a 32-bit module the OTE is formatted as follows:

```plaintext
# .lmo 'doscall1'

hmte=00a7 pmte=1f9d59f58 mflags=0698b594 c:\os2\dll\doscall1.dll

<table>
<thead>
<tr>
<th>obj</th>
<th>vsize</th>
<th>vbase</th>
<th>flags</th>
<th>ipagemap</th>
<th>cpagemap</th>
<th>hob</th>
<th>sel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>00001354</td>
<td>0a010000</td>
<td>80009025</td>
<td>00000000</td>
<td>00000000</td>
<td>0ad</td>
<td>d00e</td>
</tr>
<tr>
<td>0002</td>
<td>000002e8</td>
<td>0a020000</td>
<td>80002025</td>
<td>00000003</td>
<td>00000000d</td>
<td>00ac</td>
<td>d017</td>
</tr>
<tr>
<td>0003</td>
<td>000001844</td>
<td>0a030000</td>
<td>80001025</td>
<td>000000010</td>
<td>000000002</td>
<td>00ab</td>
<td>d01f</td>
</tr>
<tr>
<td>0004</td>
<td>000002e0</td>
<td>0a040000</td>
<td>80001025</td>
<td>000000012</td>
<td>00000001</td>
<td>00aa</td>
<td>d027</td>
</tr>
<tr>
<td>0005</td>
<td>00000540</td>
<td>0a050000</td>
<td>8000d025</td>
<td>000000013</td>
<td>000000006</td>
<td>00a9</td>
<td>d02e</td>
</tr>
<tr>
<td>0006</td>
<td>00000270</td>
<td>0a060000</td>
<td>80001023</td>
<td>000000019</td>
<td>00000001</td>
<td>00a8</td>
<td>d037</td>
</tr>
<tr>
<td>0007</td>
<td>000001b40</td>
<td>0a070000</td>
<td>80001003</td>
<td>00000001a</td>
<td>00000002</td>
<td>0000</td>
<td>d03f</td>
</tr>
</tbody>
</table>
```

The OTE fields formatted have the following meaning:

- **obj**: Object number. This is a sequential index of module object. Index entries appearing in the link-edit map will correspond with these values.
- **vsize**: (ote_size) Object virtual size
- **vbase**: (ote_base) Object base virtual address
- **flags**: (ote_flags) Attribute flags. The interpretations of these are displayed to the right of the each line. They are assigned as follows:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>.lmo msg</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJREAD</td>
<td>0x00000001</td>
<td>r</td>
<td>Readable Object</td>
</tr>
<tr>
<td>OBJWRITE</td>
<td>0x00000002</td>
<td>w</td>
<td>Writeable Object</td>
</tr>
<tr>
<td>OBJEXEC</td>
<td>0x00000004</td>
<td>x</td>
<td>Executable Object</td>
</tr>
<tr>
<td>OBJRsrc</td>
<td>0x00000008</td>
<td>rsrsc</td>
<td>Resource Object</td>
</tr>
<tr>
<td>OBJDISCARD</td>
<td>0x00000010</td>
<td>disc</td>
<td>Object is Discardable</td>
</tr>
</tbody>
</table>
OBJSHARED  0x00000020 shr  Object is Shared
OBJPRELOAD  0x00000040  prel Object has preload pages
OBJINVALID  0x00000080  inv  Object has invalid pages
OBJZEROFIL  0x00000100  zfill Object has zero-filled pages
OBJRESIDENT  0x00000200 Object is resident
OBJALIAS16  0x00001000  alias 16:16 alias required
OBJBIGDEF   0x00002000  big  Big/Default bit setting
OBJCONFORM  0x00004000  conf  Object is conforming for code
OBJIOPL     0x00008000  iopl Object I/O privilege level
OBJMADEPRIV 0x40000000 Object is made private for debug
OBJALLOC    0x80000000 Object is allocates used by the loader

ipagemap  (ote_pagemap) Object page map index.

cpagemap  (ote_mapsize) Number of entries in object page map.

hob       (ote_seghdl) Memory object handle of object data.

sel       (ote_selector) Selector assigned to this object.

If either the segment table or object is not in storage then the following message is issued:
%nnnnnnnnx - swapped

-------------------------------------------------

.M - Format Memory Structures

Format memory management structures (VMOB, VMAR, VMAL, VMCO, VP and PF).

Syntax:
.M options
   A
   O
   C
   L
   V
   P

Parameters:
A
Format Memory Arena Records (VMARs). See .MA command for more information.

O
Format Memory Object Records (VMOBs). See .MO command for more information.
C  Format Memory Context Records (VMCOs). See .MC command for more information.

L  Format Memory Alias Records (VMALs). See .ML command for more information.

V  Format Virtual Page structures (VPs). See .MV command for more information.

P  Format Page Frame structures (PFs). See .MP command for more information.

options
See the corresponding .M.r command for details of applicable options.

Prior to fix pack 29 for Warp V3, the .M command defaults to:

.MAMC

From fix pack 29 of Warp V3 and base Warp V4 the .M command is consistent for both Kernel Debugger and Dump Formatter, and defaults to .MAMC

For further details see the M option of the .MA command.

--------------------------------------------

.MA - Format Memory Arena Records (VMAR)

Display memory arena records (VMARs). Optionally format related object records (VMOBs), alias records (VMALs) and context records (VMCOs).

Syntax:

```
.MA                                                        `har
  M                 C         maddr
  A                 B         F
  A                        L     C
  A                        R     H
  A                        L     n
```

Parameters:

A

This is option is used with (and implies) the M option. It causes the a match for private area addresses to be made across all contexts. See the M option for further details.

Note:

Under Kernel Debugger the default is to match addresses in the current context only.

Under Dump Formatter address matches are made across all contexts, that is the A option is in permanent
Display in-use (busy) arena records in sequential order.

Display chained memory structures.

Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an * suffix. The related structures are:

- aliases to the associated arena record (VMALs).
- arena records of all associated alias records (VMARs).
- shared instance data objects for all related arena records
- context records for shared objects of all associated arena records (VMCOs). See .MC command.
- object records of all associated arena records (VMOBs). See .MO command.

Display free arena records.

Follow the arena hash chain pointer. The hash chain is used by virtual memory management to look up a memory object for a given context from a linear address. The algorithm proceeds as follows:

The linear address is bitwise ANDed with the hash table mask obtained from at_lHashNumbMask (VMAT+0x14). The result is shifted right by the allocation granularity for the arena. This is obtained from at_lHashNumbShift (VMAT+0x18). The result provides an index into the hash table, which is a table of arena handles that head each hash chain. The hash table address is obtained from ah_paharHash (VMAH+0x14) and the VMAT address is obtained from ah_pat (VMAH+0x18).

For OS/2 Warp V3.0, the hashing algorithm amounts to the following:

System Arena:

```
index=(linear address >> 0x0c) & 0x1ff
```

Tiled Shared and Private arenas:

```
index=(linear address >> 0x10) & 0x1ff
```

VDM Private arenas:

```
index=(linear address >> 0x0c) & 0x1ff
```

Follow the arena forward (left) chain pointer. Arena records for each arena are chained using a double-linked circular chain. The Dump Formatter or Kernel Debugger will not detect wrap-around. This option must therefore be limited by specifying a fixed number of arena records, using the Ln operand, or interrupted using Ctrl+C.

Searches for all arena records (of all contexts) that represent virtual memory that encloses the address specified in maddr. If maddr is not specified then the current CS:EIP is taken as the matching address. If the storage is in the private arena Kernel Debugger will search the current context only unless the

An address expression may be specified. A option is specified. The Dump Formatter always searches for matches in all contexts.

Follow the arena backward (right) chain pointer. Arena records for each arena are chained using a double-linked circular chain. The Dump Formatter or Kernel Debugger will not detect wrap-around. This option must therefore be limited by specifying a fixed number of arena records, using the Ln operand, or interrupted using Ctrl+C.

Specifies the matching address to be used with the M option.

Specifies the linear address of a specific arena record to be formatted.

Specifies the number of arena records to display.
**.har**

Specifies the handle of a specific arena record to be formatted.

### Results & Notes:

Arena records are in contiguous storage, which is anchored from the address given by global variable:`_parvmOne`

Output from the `.MA` command is formatted using a common template with minor variations.

**Note:** Because a common display template is used for all forms of arena record certain fields will be irrelevant to the records being viewed and may contain garbage information. Specific cases are noted in the examples where this applies.

The following are example of the nine formats of arena record:

- Free Arena Record
- Sentinel Arena Record
- Boundary Sentinel Arena Record
- System Arena Records mapped by GDT selectors
- System Arena Records not mapped by GDT selectors
- Shared Arena Records for Shared Data
- Shared Arena Records for Instance Data
- Private Arena Records for non-shared Data
- Private Arena Records for Shared Data

For a description of the fields formatted by `.MA` select `.MA Output Field Descriptions`

For more examples using of the `.M` family of commands see: *Exploring Memory Management.*

--------------------------------------------

### Free Arena Record

<table>
<thead>
<tr>
<th>har</th>
<th>par</th>
<th>cpg</th>
<th>va</th>
<th>flg</th>
<th>next</th>
<th>prev</th>
<th>link</th>
<th>hash</th>
<th>hob</th>
<th>hal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0263</td>
<td>%fef2948c</td>
<td>000294a2</td>
<td>%00320000</td>
<td>168</td>
<td>0233</td>
<td>0262</td>
<td>0000</td>
<td>0000</td>
<td>02df</td>
<td>0000</td>
</tr>
<tr>
<td>0264</td>
<td>%fef294a2</td>
<td>000294b8</td>
<td>%00000000</td>
<td>000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>0265</td>
<td>%fef294b8</td>
<td>000294ce</td>
<td>%00000000</td>
<td>000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

### Free Arena Record Display

**Notes:**

Flag bit 0x001 reset signifies a free record.

The only fields of relevance are **har, par, and cpg**.

Bit positions 0xffe of **flg** and remaining fields may contain garbage from a previous use of the record.

For a description of the fields formatted by `.MA` select `.MA Output Field Descriptions`

--------------------------------------------

### Sentinel Arena Record
Sentinel Arena Records

**Notes:**

Flag bit 0x002 set signifies a sentinel record.

**hob** is not relevant to sentinel records. (The value displayed originates from the **max** field).

For OS/2 2.1, arena record 4 is sentinel for the system arena.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Boundary Sentinel Arena Record

**Notes:**

Flag bits 0x006 set signify a boundary sentinel record.

The boundary sentinel indicates the boundary between the shared and private arena address spaces. Consequently there is only one boundary sentinel to be found in a system.

**hob** is not relevant to sentinel records. (The value displayed originates from the **max** field).

For OS/2 2.1, arena record 5 is boundary sentinel for the shared arena.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

System Arena Record Mapped by GDT

**System arena records - address space mapped by a GDT selector**
Notes:

Flag bit 0x008 set signifies a selector mapping.

va value >= that specified in the System Arena Sentinel signifies system area area record.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

System Arena Record Not Mapped by GDT

<table>
<thead>
<tr>
<th>har</th>
<th>par</th>
<th>cpg</th>
<th>va</th>
<th>flg</th>
<th>next</th>
<th>prev</th>
<th>link</th>
<th>hash</th>
<th>hob</th>
<th>hal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000e</td>
<td>%fef2613e</td>
<td>00000001</td>
<td>%fff16000</td>
<td>001</td>
<td>01d9</td>
<td>0083</td>
<td>0000</td>
<td>000f</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>000f</td>
<td>%fef26154</td>
<td>00000001</td>
<td>%fff23000</td>
<td>001</td>
<td>0010</td>
<td>0006</td>
<td>0000</td>
<td>0010</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>0010</td>
<td>%fef2616a</td>
<td>00000002</td>
<td>%fff24000</td>
<td>001</td>
<td>0011</td>
<td>000f</td>
<td>0000</td>
<td>0011</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

System arena records - address space not mapped by a GDT selector

Notes:

Flag bit 0x008 set to 0 signifies a no selector mapping.

va value >= that specified in the System Arena Sentinel signifies system area area record.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Shared Arena Record for Shared Data

<table>
<thead>
<tr>
<th>har</th>
<th>par</th>
<th>cpg</th>
<th>va</th>
<th>flg</th>
<th>next</th>
<th>prev</th>
<th>link</th>
<th>hash</th>
<th>hob</th>
<th>hal</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b2</td>
<td>%fef26f56</td>
<td>00000010</td>
<td>%1a0a0000</td>
<td>379</td>
<td>00b6</td>
<td>00b3</td>
<td>0000</td>
<td>00be</td>
<td>0000</td>
<td>hco=001ed</td>
</tr>
<tr>
<td>00b3</td>
<td>%fef26f6c</td>
<td>00000010</td>
<td>%1a090000</td>
<td>3d9</td>
<td>00b2</td>
<td>00a9</td>
<td>0000</td>
<td>00c0</td>
<td>0000</td>
<td>hco=001ee</td>
</tr>
<tr>
<td>00b4</td>
<td>%fef26f82</td>
<td>00000010</td>
<td>%1a0e0000</td>
<td>379</td>
<td>00bb</td>
<td>00b5</td>
<td>0000</td>
<td>00c1</td>
<td>0000</td>
<td>hco=0022e</td>
</tr>
</tbody>
</table>

Shared arena, shared data.

Notes:

Flag bit 0x200 set to 1 signifies shared arena, shared data.

Context records chained from hco value will list the processes that currently share the memory object represented by this arena record.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Shared Arena Record for Instance Data
Shared arena, instance data.

Notes:

Flag bit 0x200 set to 0 with a va value not in the system arena and 0 hptda indicates shared arena instance data.

Object records chained from hob value will list the objects and processes that map to the common virtual address range represented by this arena record.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Private Arena Record Non-Shared Data

Private non-shared data, process owned arena records

Notes:

Arena records not satisfying the criteria for any of the System, Sentinel or Shared Arena records are assumed to be private arena records.

If the private memory object is shared (for example, .EXE code segments running in more than one process) then the associated private arena records for the sharing processes are chained from the link field as long as hal is zero.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Private Arena Record Shared Data

Private shared data, process owned arena records

Notes:
Arena records not satisfying the above criteria are assumed to be private arena records.

If the private memory object is shared (for example, .EXE code segments running in more than one process) then the associated private arena records for the sharing processes are chained from the link field as long as hal is zero.

For a description of the fields formatted by .MA select .MA Output Field Descriptions

--------------------------------------------

Output from .MA is in a tabular format of the following form:

<table>
<thead>
<tr>
<th>har</th>
<th>par</th>
<th>cpg</th>
<th>va</th>
<th>flg</th>
<th>next</th>
<th>prev</th>
<th>link</th>
<th>hash</th>
<th>hob</th>
<th>hal</th>
<th>max</th>
<th>sel</th>
<th>hco</th>
<th>hptda</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005</td>
<td>%fef26078</td>
<td>00000040</td>
<td>00000000</td>
<td>007</td>
<td>0259</td>
<td>006e</td>
<td>0000</td>
<td>fff0</td>
<td>0000</td>
<td>max=fff0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>%fef260d0</td>
<td>00000000</td>
<td>%04000000</td>
<td>009</td>
<td>000b</td>
<td>0008</td>
<td>0000</td>
<td>0000</td>
<td>fff0</td>
<td>0000</td>
<td>sel=0120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00b4</td>
<td>%fef26f82</td>
<td>00000000</td>
<td>%1a0e0000</td>
<td>379</td>
<td>00bb</td>
<td>00b5</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>hco=0022e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00ea</td>
<td>%fef27426</td>
<td>00000000</td>
<td>%1a0e0000</td>
<td>379</td>
<td>00bb</td>
<td>00b5</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>hco=0022e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00e7</td>
<td>%fef273e4</td>
<td>00000000</td>
<td>%00020000</td>
<td>179</td>
<td>008</td>
<td>0000</td>
<td>0000</td>
<td>011e</td>
<td>0000</td>
<td>hptda=0097</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The field headings have the following meaning:

har
The handle of the arena record being formatted. This is the unique identifier by which the arena record is known.

par
The linear address of the arena record being displayed.

cpg
The size of the contiguous linear address range reserved by this arena record expressed as a hexadecimal number of pages. This value will be greater then or equal to the size of the memory object that occupies this linear address range.

For small allocations (<64K) in non-system arenas this will usually be rounded to 0x10 pages so that the CRMA may be applied.

For free records this field is used as a chain pointer to the next free record but only the low order 20 bits are displayed by .MA.

va
The linear address of the beginning of the memory object represented by the arena record.

flg
Arena record flags. These may take a combination of the following settings:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_INUSE</td>
<td>0x001</td>
<td>Record not on free list</td>
</tr>
<tr>
<td>AR_TAG</td>
<td>0x006</td>
<td>Record type mask</td>
</tr>
<tr>
<td>AR_TAGREG</td>
<td>0x000</td>
<td>Regular record</td>
</tr>
<tr>
<td>AR_TAGSEN</td>
<td>0x002</td>
<td>Sentinel</td>
</tr>
<tr>
<td>AR_TAGSEN</td>
<td>0x002</td>
<td>Boundary sentinel</td>
</tr>
<tr>
<td>AR_SELMAP</td>
<td>0x008</td>
<td>Memory mapped by selector</td>
</tr>
<tr>
<td>AR_SELBASEALL</td>
<td>0x00c</td>
<td>Base selector map all</td>
</tr>
<tr>
<td>AR_SELMASK</td>
<td>0x00c</td>
<td>Selector map mask</td>
</tr>
<tr>
<td>AR_RELOAD</td>
<td>0x010</td>
<td>Pre-reserved for huge item or or reserved for reload of MTE object</td>
</tr>
</tbody>
</table>
AR_WRITE       0x020  write permission
AR_USER        0x040  user permission
AR_EXEC        0x080  executable permission
AR_READ        0x100  Read permission
AR_HCO         0x200  Record linked to Context List
AR_GUARD       0x400  guard page
AR_SGS         0x800  Registered under Screen Group

Switch control.
next
Handle of next arena record within the same arena (private, shared or system).
prev
Handle of the previous record within the same arena (private, shared or system).
link
Handle of an associated arena record.
  For private arena sentinel records this points to the boundary sentinel.
  For shared arena shared data this points to other private arenas sharing the same
  object.
  For alias objects this points to the arena record of the associated (aliased) object.
hash
Handle of the next arena record whose va hashes to the same hash chain.
hob
The handle of the associated memory object record. See .MO command.
hal
The handle of the associated memory alias record. See .ML command. If this field is set to a value of 0xffff then this
is not the handle of an alias record, but signifies that this arena has been privatized by the creation of an alias to it.
hptda=hhhh
The handle of the pseudo-object that is the PTDA of the process that has this arena record assigned to its private
address space. Use .MO hhhh to display the PTDA pseudo-object and hence obtain the address of its virtual
address.
max=%mmmmmmmm
Maximum linear address of the area headed by this sentinel record.
set=ssss
GTD selector that is assigned to a system arena memory object.
hco=cccc
Handle of the first context record that represents processes sharing this shared arena, shared data memory object.
  See .MC command.

--------------------------------------------
.MC - Format Memory Context Records (VMCO)

[icon]
Display memory context records (VMCOs).

Syntax:
Parameters:

B
Display in-use (busy) alias records in sequential order.

C
Display chained context records.

Chaining causes related context records that are chained from hconext of the current context to be formatted. The head of each group indicated by an * suffix. Context records are chained to represent each instance of an object being shared among several processes. The head of the chain is pointed to by the hco field of the corresponding arena record. See .MA command for information on formatting arena records.

Notes:

There is no pointer to the arena record from the context record. Associated arena records have to be found by scanning arena or object records.

The C option will not format context records from the head of the chain. Do achieve this, locate the corresponding arena record and use

.FAC bar

F
Display free alias records.

laddr
Specifies the linear address of a specific context record to be formatted.

Ln
Specifies the number of context records to display.

hco
Specifies the handle of a specific context record to be formatted.

Results & Notes:

Context records are in contiguous storage, which is anchored from the address given by global variable:

_pcovmOne

Output from the .MC command appears in one of two formats:

Free Context Records
Busy Context Records

For a description of the fields formatted by .MC select .MC Output Field Descriptions.

For more examples using of the .M family of commands see: Exploring Memory Management.

In some versions prior to fix pack 29 for Warp V3 the Dump Formatter did not interpret and format the VMCO correctly. In addition VCMO chains were not followed correctly when the C options was used. These problems are fixed from fix pack 29 of Warp V3 and base Warp V4.

--------------------------------------------

Free Alias Records

hco=00313  pco=ffe4bf7a  pconext=ffe4bf7f
Free Context Record Display

Notes:

pconext is used as a chain pointer to the next free context record.

For a description of the fields formatted by .MC select .MC Output Field Descriptions

--------------------------------------------

Busy Context Records

hco=00001 pco=ffe4b020 hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
hco=00002 pco=ffe4b025 hconext=00000 hptda=00ae f=13 pid=0002 c:pmshell.exe
hco=00003 pco=ffe4b02a hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
hco=00004 pco=ffe4b02f hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe

Selector Busy Context Records

Notes:

Flag bit 0x20 set signifies a context handle > 64k. In effect this is a 1 bit extension of the 16 bit hco field of the VMCO. The .ML command takes this into account when formatting VMCO records.

Flag bit 0x80 set signifies that the context has been privatized. This implies that the object was originally shared but a private instance of it has subsequently been created. Typically this occurs when DosDebug is used to access a debugee's shared data.

For a description of the fields formatted by .MC select .MC Output Field Descriptions

--------------------------------------------

.MC Output Field Descriptions

Output from .MC appears in one of is of the following forms:

hco=00317 pco=ffe4bf8e pconext=ffe4bf93
hco=00001 pco=ffe4b020 hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe

Each of the fields has the following meaning:

hco= The handle of the context record being formatted. This is the unique identifier by which the context record is known.

pco= The linear address of the context record being formatted.

hconext= For busy records this is the handle of the next context record that represents another user of the related (shared) object. For free records this is the handle of the next free record.
The handle of the PTDA pseudo-object that represents the process sharing the corresponding memory object.

Context record flags.

The following flags are defined:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO_CREATOR</td>
<td>0x01</td>
<td>originating context</td>
</tr>
<tr>
<td>CO_PRIV</td>
<td>0x80</td>
<td>Privatized context</td>
</tr>
<tr>
<td>CO_HCOH</td>
<td>0x20</td>
<td>Next context record handle &gt; 64k</td>
</tr>
<tr>
<td>CO_WRITE</td>
<td>0x02</td>
<td>Write access</td>
</tr>
<tr>
<td>CO_USER</td>
<td>0x04</td>
<td>User attribute</td>
</tr>
<tr>
<td>CO_EXEC</td>
<td>0x08</td>
<td>Execute access</td>
</tr>
<tr>
<td>CO_READ</td>
<td>0x10</td>
<td>Read access</td>
</tr>
<tr>
<td>CO_GUARD</td>
<td>0x40</td>
<td>Guard page</td>
</tr>
</tbody>
</table>

This names the process Id and process executable that corresponds to the hptda field.

* EBOOKIE (IPFTAGS.DEF)

.MK - Display Memory Lock Information Records (VMLKI)

Display memory lock information records.

Note:

This command was implemented by feature 82818 for the ALLSTRICT and HSTRICT kernels only. It is not available in either of the GA versions for OS/2 Warp V3.0 or OS/2 V2.11.

After OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 .MK may be enabled under the RETAIL kernel by means of the LOCKS option of the RASKDATA CONFIG.SYS statement.

Under the ALLSTRICT and HSTRICT kernels .MK may be disabled by specifying the NLOCKS option of the RASKDATA statement.

Syntax:

```
.MK
```

Parameters:

none

Lists all lock information records for all memory objects with locked records.

Ln

Specifies the number of lock information records to display for a given hob.
Specifies the handle of a specific object record whose lock information records are to be formatted.

**Results & Notes:**

Lock information records are maintained for outstanding memory locks in memory lock information records (VMLIs) which are located at the address given by global variable:

```
_pVMLIHead
```

When a memory lock request is successfully executed a lock handle is returned to the caller for later use when unlocking memory. The lock handle normally resides in the caller's storage. It comprises a concatenation of:

- the requestor's hptda
- the hob whose pages are being locked
- the page number
- the number of pages
- request flags

In addition a check-sum or signature is calculated from these values and stored with the lock handle.

The VMLI is a copy of the constituents of the lock handle that resides in system memory. In addition it includes:

- the requestor's return address
- a pointer to the next VMLI
- a pointer to the requestor's lock handle

The .MK command formats the contents of the VMLI then re-calculates the signature. The calculated and saved signatures should be identical.

Next the lock handle is accessed. If it differs from the corresponding VMLI then it too is formatted and the signature is re-calculated and displayed. If either the formatted lock handle and corresponding VMLI or the calculated and extracted signatures disagree then a problem may be indicated. For example, an overlayed or freed lock handle. However, there is no requirement for lock requestors to retain their lock handles in their original locations.

**Warning:**

Prior to fix pack 29 for Warp V3, the Kernel Debugger can trap when attempting to format lock handles from freed memory. This is fixed in the Kernel Debugger and incorporated into the Dump Formatter in defect 155843 from fix pack 29 for Warp V3 and base Warp V4.

**Note:**

When feature 82818 is present VMLI records are automatically formatted when displaying memory object records with locked pages, using the .MO command.

Output from the .MK command appears as follows:

```
##.mk

<table>
<thead>
<tr>
<th>pvmli</th>
<th>cs</th>
<th>eip</th>
<th>phlock</th>
<th>cpg</th>
<th>va</th>
<th>flg</th>
<th>hptda</th>
<th>hob</th>
<th>sig</th>
<th>csig</th>
</tr>
</thead>
<tbody>
<tr>
<td>%fe679f1c</td>
<td>0170</td>
<td>fffa015d</td>
<td>%fd17d480</td>
<td>0001</td>
<td>%013f1000</td>
<td>0003</td>
<td>0091</td>
<td>0424</td>
<td>18aa</td>
<td></td>
</tr>
<tr>
<td>%fe681ad0</td>
<td>0170</td>
<td>fffa015d</td>
<td>%fd17d468</td>
<td>0001</td>
<td>%013f1000</td>
<td>0003</td>
<td>0091</td>
<td>0424</td>
<td>18aa</td>
<td></td>
</tr>
<tr>
<td>%fe745393</td>
<td>0170</td>
<td>fff3e551</td>
<td>%ffe006ff</td>
<td>0001</td>
<td>02ff</td>
<td>0016</td>
<td>0252</td>
<td>0252</td>
<td>01ff</td>
<td></td>
</tr>
<tr>
<td>%fe712c54</td>
<td>0170</td>
<td>fff3e551</td>
<td>%ffe00577</td>
<td>0003</td>
<td>02ff</td>
<td>0016</td>
<td>0252</td>
<td>0252</td>
<td>01ff</td>
<td></td>
</tr>
<tr>
<td>%fe761e0c</td>
<td>0908</td>
<td>00000878</td>
<td>%7b6b7d0c</td>
<td>0001</td>
<td>%ffee9000</td>
<td>0005</td>
<td>0091</td>
<td>0190</td>
<td>01ff</td>
<td></td>
</tr>
<tr>
<td>%fe777e18</td>
<td>0908</td>
<td>000008a1</td>
<td>%7b6b7d0c</td>
<td>0006</td>
<td>%ffeea000</td>
<td>0005</td>
<td>0091</td>
<td>0227</td>
<td>01bc</td>
<td></td>
</tr>
<tr>
<td>%fe777e3c</td>
<td>0908</td>
<td>00000809</td>
<td>%7b6b7d0c</td>
<td>0001</td>
<td>%ffee0000</td>
<td>0005</td>
<td>0091</td>
<td>022c</td>
<td>01bc</td>
<td></td>
</tr>
<tr>
<td>%fe777e60</td>
<td>0908</td>
<td>0000072b</td>
<td>%7c224066</td>
<td>0002</td>
<td>%17c4000</td>
<td>0005</td>
<td>0091</td>
<td>0199</td>
<td>07e7</td>
<td></td>
</tr>
<tr>
<td>%fe777e84</td>
<td>0908</td>
<td>0000072b</td>
<td>%7c224058</td>
<td>0001</td>
<td>%7a022000</td>
<td>0001</td>
<td>0091</td>
<td>0168</td>
<td>a224</td>
<td></td>
</tr>
<tr>
<td>%fe777ef8</td>
<td>0908</td>
<td>000006ee</td>
<td>%7c22403c</td>
<td>0001</td>
<td>%7a022000</td>
<td>0001</td>
<td>0091</td>
<td>0168</td>
<td>a224</td>
<td></td>
</tr>
<tr>
<td>%fe777ef28</td>
<td>0908</td>
<td>000006ee</td>
<td>%7c22404a</td>
<td>0002</td>
<td>%17c8000</td>
<td>0005</td>
<td>0091</td>
<td>0196</td>
<td>7eaf</td>
<td></td>
</tr>
</tbody>
</table>
```
The field headings have the following meaning:

**pvml**
Address of the VMLI record.

**cs**
Code selector of the requestor of the memory locking function. For calls made through a DevHlp request this is taken from TCBpDHRetAddr (TCB + 0x74). For internal requests the immediate caller of _VmLockMem is displayed.

A blank value indicates information from the lock handle is being formatted, because it does not agree with the corresponding VMLI. See note above.

If the lock request was made by a 16-bit caller then the 16-bit far return address is contained within the eip only and the cs value is the next word from the stack following the return address.

**eip**
The instruction pointer of the requestor of the memory locking function. For calls made through a DevHlp request this is taken from TCBpDHRetAddr (TCB + 0x74). For internal requests the immediate caller of _VmLockMem is displayed.

A blank value indicates information from the lock handle is being formatted, because it does not agree with the corresponding VMLI. See note above.

**phlock**
The address of the lock handle buffer supplied by with the lock request.

**cpg**
The number of contiguous pages locked.

**va**
The linear address of the first page locked.

**flg**
The flags saved from the lock request.

The following bit settings are defined:

<table>
<thead>
<tr>
<th>Bit value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>Lock is a long-term</td>
</tr>
<tr>
<td>0x02</td>
<td>Verify lock call</td>
</tr>
<tr>
<td>0x04</td>
<td>Lock originated from a DevHlp</td>
</tr>
</tbody>
</table>

**hptda**
The hptda of the lock requestor.

**hob**
The handle of the associated memory object record, some of whose pages are. See the .MO command.

**sig**
The signature value extracted from the VMLI or lock handle.

**csig**
The recalculated signature based on information saved in the VMLI or lock handle.

For related information see also the Virtual Memory Lock Trace.

--------------------------------------------

*ML - Format Memory Alias Records (VMAL)*
Display memory alias records (VMALs). Optionally format related arena records (VMARs), object records (VMOBs) and context records (VMCOs).

**Syntax:**

```
.ML B F hal
C laddr Ln
```

**Parameters:**

- **B**
  - Display in-use (busy) alias records in sequential order.

- **C**
  - Display chained memory structures.
    - Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an * suffix.
    - The related structures are:
      - the associated arena record (VMAR). See `.MA command`
      - aliases to the associated arena record (VMALs).
      - arena records of all associated alias records
      - shared instance data objects for all related arena records
      - context records for shared objects of all associated arena records (VMCOs). See `.MC command`
      - object records of all associated arena records (VMOBs). See `.MO command`

- **F**
  - Display free alias records.

- **maddr**
  - Specifies the matching address to be used with the **M** option.

- **laddr**
  - Specifies the linear address of a specific alias record to be formatted.

- **Ln**
  - Specifies the number of arena records to display.

- **hal**
  - Specifies the handle of a specific alias record to be formatted.

**Results & Notes:**

Alias records are in contiguous storage, which is anchored from the address given by global variable:

```
_palVMAliases
```

From fix pack 29 for Warp V3 and base Warp V4 the Dump Formatter .ML command has been enhanced to work independently of kernel symbols. Prior to these releases, kernel symbols were required to enable the Dump Formatter to locate `_palVMAliases`.

Output from the `.ML` command appears in one of three formats.

- Free Alias Record
- Selector Alias Record
- Linear Address Alias Record

For a description of the fields formatted by `.ML` select `.ML Output Field Descriptions`
Free Alias Records

<table>
<thead>
<tr>
<th>hal</th>
<th>pal</th>
<th>palnext</th>
<th>pgoff</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>000e</td>
<td>fffe61088</td>
<td>ffe610a0</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>000f</td>
<td>fffe61090</td>
<td>ffe61088</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>fffe61098</td>
<td>ffe61090</td>
<td>d4460</td>
<td>0</td>
</tr>
</tbody>
</table>

Free Alias Record Display

Notes

- Flag bit 0x001 reset signifies a free record.
- The only fields of relevance are hal, pal, and palnext.

Selector Alias Records

<table>
<thead>
<tr>
<th>hal</th>
<th>pal</th>
<th>har</th>
<th>cs</th>
<th>ds</th>
<th>cref</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>000a</td>
<td>fffe61068</td>
<td>0208</td>
<td>0056</td>
<td>d446</td>
<td>001</td>
<td>13</td>
</tr>
<tr>
<td>000b</td>
<td>fffe61070</td>
<td>020b</td>
<td>0056</td>
<td>d446</td>
<td>001</td>
<td>13</td>
</tr>
</tbody>
</table>

Selector Alias Record Display

Notes

- Flag bit 0x02 set signifies a CS alias record.
- Flag bit 0x10 set signifies that DS is valid, i.e. CS is an alias for the same storage mapped by DS. For example after use of DosCreateCSAlias. This flag distinguishes this form of the alias record.
- This alias applies to selectors within a specific (process) context.

Linear Address Alias Records

<table>
<thead>
<tr>
<th>hal</th>
<th>pal</th>
<th>har</th>
<th>hptda</th>
<th>pgoff</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>fffe61020</td>
<td>00b8</td>
<td>009f</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0002</td>
<td>fffe61028</td>
<td>00b9</td>
<td>009f</td>
<td>0000</td>
<td>0</td>
</tr>
</tbody>
</table>
Notes

Flag bit 0x10 reset distinguishes this form of the alias record.

hptda and har refer to the context and arena which has been aliases.

The context and arena of the creator of the alias may be shown using .MLC hal.

For a description of the fields formatted by .ML select .ML Output Field Descriptions

--------------------------------------------

.ML Output Field Descriptions

Output from .ML appears in one of is of the following forms:

hal=0006 pal=ffe61048  har=01b8  hptda=009f  pgoff=00000  f=021
hal=000a pal=ffe61068  har=0208  cs=0056  ds=d446  cref=001  f=13
hal=0011 pal=ffe610a0  palnext=ffe610a8  pgoff=00000  f=000

Each of the fields has the following meaning:

hal
The handle of the alias record being formatted. This is the unique identifier by which the alias record is known.

har
The handle of the arena record which represents the aliasing linear address range. The arena record for the aliased linear address is pointed to by the link field of the aliasing har.

hptda
The handle of the PTDA object of the context which has been aliased. This may also take the value of a system owner when memory is in the process of being freed.

pgoff
The page offset into the arena which is aliased. Note: aliases may map partial objects. The number of pages aliased may be determined from the arena record which represents the aliased memory. Use .MLC /ha/ to display this information.

cs
The Code Selector created to alias R/W memory.

ds
The Data Selector which has been aliased by a Code selector.

cref
The reference count for the number of time a Code Selector alias has been requested for a given Data Selector.

palnext
The pointer to the next free alias record.

f
Alias record flags. For Selector Aliases this is a 6-bit field, for other aliases this is a 12-bit field.
The following flags are defined:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL_ISBUSY</td>
<td>0x1</td>
<td>Set if record is busy</td>
</tr>
<tr>
<td>AL_CSALIAS</td>
<td>0x2</td>
<td>Set is CS alias record</td>
</tr>
<tr>
<td>AL_MEMMAP</td>
<td>0x4</td>
<td>Set if MemMapAlias Record</td>
</tr>
<tr>
<td>AL_DBGALIAS</td>
<td>0x8</td>
<td>Set if Debug alias</td>
</tr>
<tr>
<td>AL_CSDSVALID</td>
<td>0x10</td>
<td>Set if DS selector valid</td>
</tr>
<tr>
<td>AL_DEVHLP</td>
<td>0x20</td>
<td>Set if Devhlp Alias</td>
</tr>
<tr>
<td>AL_PRIV</td>
<td>0x40</td>
<td>Set if privatized alias</td>
</tr>
<tr>
<td>AL_VDM</td>
<td>0x80</td>
<td>Set if VMD alias</td>
</tr>
<tr>
<td>AL_NOALIAS</td>
<td>0x100</td>
<td>Set if UVIRT mapping in VDMs</td>
</tr>
</tbody>
</table>

.MO - Format Memory Object Records (VMOB)

Display memory object records (VMOBs). Optionally format related alias records (VMALs), arena records (VMARs) and context records (VMCOs).

If feature 82818 is installed, then under the Kernel Debugger only, lock information records will be formatted whenever a memory object with locked pages is displayed. See the .MK command for more information.

Syntax:

```
.MO V M maddr
.B C hob
.F laddr L n
.N P S
```

Parameters:

B
Display in-use (busy) object records in sequential order.

C
Display chained memory structures.

Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an * suffix. The related structures are:

- the associated arena record (VMAR). See .MA command
- aliases to the associated arena record (VMALs).
- arena records of all associated alias records
- shared instance data objects for all related arena records
context records for shared objects of all associated arena records (VMCOs). See .MC command.

object records of all associated arena records (VMOBs).

Note: Pseudo-objects have no related memory objects.

F
Display free object records.

M
Searches for a pseudo-object whose address matches the address specified for \textit{maddr}. If no match is found then nothing is displayed.

\textbf{Notes:}

- The pseudo-object address specified must be an exact match for hit.
- The pseudo-object address is that of the object itself and not the VMOB that represents it.
- A \textit{selector:offset} form of address may not be specified. However a physical address may be specified in order to bypass virtual address validation done by Kernel Debugger and Dump Formatter.

N
Specifies that only normal object records be displayed. These are objects whose linear address allocation is represented by an arena record. Contrast this with Pseudo-Object and System Object. See also .MA command for details of arena record display.

P
Specifies that pseudo-object records be displayed.

S
Specifies that objects to be displayed are those whose memory management semaphore is busy or wanted. The memory management semaphore is used internally for serialising access to memory management structures. It should not be confused with the memory locking as provided by the DevHip_Lock, DevHip_Unlock, DevHip_VMLocl and DevHip_VMUnlock calls.

V
Specifies verbose mode of display. The address of the VMOB structure is displayed but object description and owner interpretation is suppressed.

\textit{maddr}
Specifies the matching address to be used with the \textit{M} option.

An \textit{address expression} may be specified.

\textit{laddr}
Specifies the linear address of a specific object record to be formatted.

An \textit{address expression} may be specified.

\textit{Ln}
Specifies the number of object records to display.

\textit{hob}
Specifies the handle of a specific object record to be formatted according to the criteria specified by the other options.

\textbf{Results & Notes:}

Object records are located in contiguous storage, which is anchored from the address given by global variable: \_pobvmOne

Output from the .MO. command appears in one of four formats:

- Normal Object
- Pseudo-Object
- Free Object Record
- System Object

From fix pack 29 for Warp V3 and base Warp V4 the output of this command has been enhanced by adding \texttt{hmte} interpretation for system
objects. The appears on the right hand end of the command output following the textual interpretation of the own field.

From fix pack 35 for Warp V3 and base Warp V4 Device Driver and File System Driver allocated objects will record the driver's true hmte as its hmte owner Id instead of one of the system objects dd1-dd16 and fsd1-fsd8.

Prior to fix pack 29 for Warp V3 the .MO command could trap D while formatting MTE objects. This is fixed from fix pack 29 for Warp V3 and base Warp V4.

For a description of the fields formatted by .MO select .MO Output Field Descriptions

For more examples using of the .M family of commands see: Exploring Memory Management.

--------------------------------------------

Normal Object Records

<table>
<thead>
<tr>
<th>hob</th>
<th>har</th>
<th>hobnxt</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>000d</td>
<td>000c</td>
<td>0000</td>
<td>0325</td>
<td>ffb8</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 lock</td>
</tr>
<tr>
<td>000e</td>
<td>000d</td>
<td>0000</td>
<td>0000</td>
<td>ffaa</td>
<td>0006</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 os2krnl</td>
</tr>
<tr>
<td>000f</td>
<td>000e</td>
<td>0000</td>
<td>0000</td>
<td>ffaa</td>
<td>0006</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 os2krnl</td>
</tr>
<tr>
<td>0010</td>
<td>008c</td>
<td>0000</td>
<td>000e</td>
<td>0115</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 priv 0002 cpmshell.exe</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>0010</td>
<td>0000</td>
<td>0000</td>
<td>ffa37</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 rdata</td>
</tr>
</tbody>
</table>

Normal Object Record Display

<table>
<thead>
<tr>
<th>hob</th>
<th>pob</th>
<th>har</th>
<th>hobnxt</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>%fec80020</td>
<td>0001</td>
<td>fec8</td>
<td>0000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0002</td>
<td>%fec80030</td>
<td>0002</td>
<td>fec8</td>
<td>0000</td>
<td>ffe3</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0003</td>
<td>%fec80040</td>
<td>0003</td>
<td>fec8</td>
<td>0000</td>
<td>ffec</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>

Normal Object Record Display - verbose form

Notes

The verbose form is specified using the V option. This causes the suppression of owner and hmte interpretation.

For a description of the fields formatted by .MO select .MO Output Field Descriptions

--------------------------------------------

Pseudo-Object Records

<table>
<thead>
<tr>
<th>hob</th>
<th>va</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004</td>
<td>%ff013238</td>
<td>8000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 vmah</td>
</tr>
<tr>
<td>0005</td>
<td>%ff013190</td>
<td>8000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 vmah</td>
</tr>
<tr>
<td>0006</td>
<td>%ff06a891</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte dscalls.dll</td>
</tr>
<tr>
<td>0072</td>
<td>%ffe3c7d4</td>
<td>8000</td>
<td>ffe8</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte ptda 0001 *sysinit</td>
</tr>
<tr>
<td>007a</td>
<td>%ff0b26fa</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte mdm.dll</td>
</tr>
<tr>
<td>007b</td>
<td>%ff0b26b8</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte fshelper.dll</td>
</tr>
<tr>
<td>0091</td>
<td>%ffe73499ac</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte cpmshapim.dll</td>
</tr>
<tr>
<td>0098</td>
<td>%b9e4060</td>
<td>8000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 vmah</td>
</tr>
<tr>
<td>009d</td>
<td>%fe722fb8</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 mte c:clock02.sys</td>
</tr>
</tbody>
</table>
Pseudo-Object Record Display

Notes

The 0x8000 flag bit signifies as pseudo-object.

For a description of the fields formatted by .MO select .MO Output Field Descriptions

Free Object Records

```
02b5  %fec82b70  0000 0000 0000  0000 00  00 00 00 free
02b6  %fec82b80  0000 0000 0000  0000 00  00 00 00 free
02b7  %fec82b90  0000 0000 0000  0000 00  00 00 00 free
02b8  %fec82ba0  0000 0000 0000  0000 00  00 00 00 free
02b9  %fec82bb0  0000 0000 0000  0000 00  00 00 00 free
```

Free Object Record Display

Notes

Flag bit 0x001 reset signifies a free record.

The only fields of relevance are va and pob when the V option is specified.

The va field is used a link field to other free VMOBs.

For a description of the fields formatted by .MO select .MO Output Field Descriptions

System Object IDs

```
fff0 vmllock
fff1 vmob
fff2 vmsgs
fff3 vmbmp16
```

Example System Object Display

Notes

System object IDs are not represented by VMOB structures. They are pre-defined IDs for system components.

The Dump Formatter and Kernel Debugger display only object names when displaying system objects.
## .MO Output Field Descriptions

Output from .MO appears tabular for with one of the following headings:

<table>
<thead>
<tr>
<th>hob</th>
<th>va</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004</td>
<td>%fff13238</td>
<td>8000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0006</td>
<td>%fff0a891</td>
<td>8000</td>
<td>ffa6</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hob</th>
<th>pob</th>
<th>har</th>
<th>hobnxt</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0003</td>
<td>%fec80040</td>
<td>0003</td>
<td>fec8</td>
<td>0000</td>
<td>ffe3</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0004</td>
<td>%fec80050</td>
<td>%fff13238</td>
<td>8000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hob</th>
<th>har</th>
<th>hobnxt</th>
<th>flgs</th>
<th>own</th>
<th>hmte</th>
<th>sown,cnt</th>
<th>lt</th>
<th>st</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0001</td>
<td>fec8</td>
<td>0000</td>
<td>ffe1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0002</td>
<td>0002</td>
<td>fec8</td>
<td>0000</td>
<td>ffe3</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

Each of the heading fields has the following meaning:

- **hob**: The handle of the object record being formatted. This is the unique identifier by which the object record is known.
- **hobnext**: The handle of next shared instance data object that maps to the same linear address range (shares the same arena record but maps to a different physical address).
- **har**: The handle of the arena record that describes the linear address range allocated to this object.
- **pob**: The linear address of the object record.
- **va**: The virtual address of the pseudo-object named in the object description.
- **flgs**: Object record flags.

The following flags are defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB_PSEUDO</td>
<td>0x8000</td>
<td>Pseudo-object</td>
</tr>
<tr>
<td>OB_API</td>
<td>0x4000</td>
<td>API located object</td>
</tr>
<tr>
<td>OB_LOCKWAIT</td>
<td>0x2000</td>
<td>Waiting for a lock conflict to resolve</td>
</tr>
<tr>
<td>OB_LALIAS</td>
<td>0x1000</td>
<td>Object has aliases</td>
</tr>
<tr>
<td>OB_SHARED</td>
<td>0x0800</td>
<td>Object's contents are shared</td>
</tr>
<tr>
<td>OB_UVIRT</td>
<td>0x0400</td>
<td>UVirt object</td>
</tr>
<tr>
<td>OB_ZEROINIT</td>
<td>0x0200</td>
<td>Object is zero-initialised</td>
</tr>
<tr>
<td>OB_RESIDENT</td>
<td>0x0100</td>
<td>Initial allocation was resident</td>
</tr>
<tr>
<td>OB_LOWMEM</td>
<td>0x0040</td>
<td>Object is in low memory</td>
</tr>
<tr>
<td>OB_GUARD</td>
<td>0x0080</td>
<td>Guard page attribute</td>
</tr>
<tr>
<td>OB_READ</td>
<td>0x0010</td>
<td>Readable attribute</td>
</tr>
<tr>
<td>OB_USER</td>
<td>0x0008</td>
<td>User attribute</td>
</tr>
</tbody>
</table>
OB_WRITE        0x0004  Writeable attribute
OB_HUGE         0x0002  Object is huge
OB_SHRINKABLE   0x0001  Object is Shrinkable (only if also
                         OB_SHARED)
OB_DHSETMEM     0x0001  DevHlp_VMSetMems are allowed the object

Notes:

See Pseudo-Objects when OB_PSEUDO is set.

OB_API is set as a result of allocation made by some APIs (for example, DosExecPgm). It forces page
allocation and signals a likelihood of long-term locking.

OB_HUGE is set when the object is created by DosAllocHuge API.

When OB_LOCKWAIT is set then the thread has detected a lock request conflict and wishes to wait for the
conflict to resolve. The conflict occurs because a contiguous storage lock has been requested but cannot be
satisfied because one or more of the pages are already short-term locked. If the current request is for a
short-term lock then the thread will wait up to 10 seconds for the conflict to clear. If the time-out expires then
the current short-term lock request ends in error and the following message appears on the debugger
screen:

VMLOCK: Short term lock for > 10 secs: hob=\hsb

If the current request is for a long-term lock then the thread will wait indefinitely. In both cases the block Id
the thread waits on is the address of the VMOB flag word (VMOB+0x4). See .PB command for information
on thread slots waiting on Block Ids.

own

This is the hob of the owner of this object. The owning hob may be in one of three categories:

1. System Owner
   Used to indicate system owned objects. The owner description usually indicates
   the type of object that is being displayed. For example, the LDT for process 9
   running pulse.exe is owned by system object 0xffb9 and has a description
   ldt  0009 c:pulse.exe.

2. Module Owner (hmte)
   This is used for shared (code and data) objects that are part of a load module file.
   The .hmte of the associated load module is used as the Owner Id for the object.
   The object description names the owning module from the MTE/SMTE structures.
   See .LM command for related information.

3. Process owner (hptda)
   Process owned objects are those allocated in the private arena of a running
   process and are either dynamic allocations or non-shared module segments, for
   example stack segments. The hptda of the process is used as the owner. The
   owner description names the process id and executable module of the owning
   process.

hmte

This names the hmte or System Object Id of the executable code that allocated the memory object or in the case of
data segments of modules, with which they are associated.

sown

Semaphore owner id. This is the thread slot number that owns the memory management semaphore associated with
this object. Memory management uses the address of the VMOB as the block Id to sleep on when the semaphore is
held. This semaphore is used to serialise access to a VMOB structure. See .PB command for information on thread
slots waiting on Block Ids.

cnt

Count of owners of the VMOB semaphore and wait flag.

The low order bit of cnt is used as a wait indicator. The high order 7 bits are a count of the number of times the
owning thread has requested the VMOB semaphore without releasing it. See sown filed above for related
Extra flags.

The following flags are defined:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMDB_SLOCK_WAIT</td>
<td>0x01</td>
<td>Waiting on short term locks to clear</td>
</tr>
<tr>
<td>VMOB_LLOCK_WAIT</td>
<td>0x02</td>
<td>Waiting on long term locks to clear</td>
</tr>
<tr>
<td>VMOB_DISC_SEG</td>
<td>0x04</td>
<td>Object is part of a discardable seg</td>
</tr>
<tr>
<td>VMOB_HIGHMEM</td>
<td>0x08</td>
<td>Object was allocated via dh_vmalloc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>using the VMDHA_USEHIGHMEM flag</td>
</tr>
</tbody>
</table>

Notes:

The lock wait flags indicate that a thread is waiting for locked pages in the memory object to be unlocked, but not to resolve a conflicting lock request: that is indicated with the OB_LOCKWAIT flag.

If a thread blocks waiting for long-term locks to clear then the address of the long-term lock count (VMOB + 0xd) is used as the Block-Id the thread blocks on. The thread blocks indefinitely.

If a thread blocks waiting for short-term locks to clear then the address of the short-term lock count (VMOB + 0xe) is used as the Block-Id the thread blocks on. The thread will block for up to 10 seconds. If after that time the short-term lock has not been cleared then an error is returned and under the debug kernel the following message is sent to the debug console:

VMLOCK: Short term lock for > 10 secs: hob=hob

See .PB command for information on thread slots waiting on Block Ids.

lt

Count of active long-term lock holders. A non-zero value indicates one or more pages of the memory object have been long-term locked, that is prevented from being paged out from physical storage. Long-term locks are expected to be held for a relatively long period of time, in the order of seconds. See .MP command for information on displaying physical storage status. See also VM Lock Trace Kernel Debugger facility.

description

The object description appears to the right of the tabular display. It is combines the interpretation of own and hmte fields. The following forms are possible:

Process owned objects

These appear as: priv pid process

where:

pid

process

is the owning process id

is the executable running the owning process

MTE Owned objects

These appear as: shared module
where:

**module**

is the name of the module that contains the object (hob) displayed.

**PTDA Pseudo-objects**

These appear as:

`ptda pid process`

where:

**pid**

is the process id in which the object is located.

**process**

is the executable running the owning process.

**MTE Pseudo-objects**

These appear as:

`mte module`

where:

**module**

is the module name that corresponds to the MTE pointed to by the va.

**LDT occupying storage**

This appears as:

`ldt pid process`

where:

**pid**

is the id of the process that owns the LDT related to the object.

**process**

is the executable running the owning process.

**Free objects**

These appear as:

`free`

**System Owned Objects**

These appear as:

`owner user`

where:

**owner**

is the system object name corresponding to the own field.

**user**

is the system object name corresponding to the hmte field.
.MP - Format Memory Page Frame Structures (PFs)

Display memory Page Frame Structures (PFs).

Syntax:

```
.MP B         F           frame
             I           laddr         Ln
             L
             R
```

Parameters:

B
Display in-use (busy) Page Frame Structures in sequential order.

Note: In-Use PFs are signified by the PF_FREE flag being reset and not by the PF_BUSY flag being set.

F
Display free Page Frame Structures.

I
Display idle Page Frame Structures. (This option was introduced at fix pack 29 of Warp V3 and base Warp V4.)

L
Follow left (forward) chain pointer. This is only of relevance to Free and Idle Page Frame Structures since these are linked in a double linked chain.

Warning: Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly. In particular the 2 high order digits of the frame number are truncated. Use this option advisedly! This is fixed from fix pack 29 for Warp V3 and base Warp V4. See Free Page Frame Structures or Idle Page Frame Structures for information on locating Idle and Free PF chains.

Note: From fix pack 29 of Warp 3.0 and GA Warp 4.0 the hmte of all objects with a system owner id is also formatted in the USER part of the description.
Follow right (backward) chain pointer. This is only of relevance to Free and Idle Page Frame Structures since these are linked in a double linked chain.

Warning: Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly. In particular the 2 high order digits of the frame number are truncated. Use this option advisedly! This is fixed from fixpack 29 for Warp V3 and base Warp V4. See Free Page Frame Structures or Idle Page Frame Structures for information on locating idle and free PF chains.

\texttt{laddr}

Specifies the linear address of a specific Page Frame Structure to be formatted.

An address expression may be specified.

\texttt{L^n}

Specifies the number of Page Frame Structures to display from the starting criterion.

\texttt{frame}

Specifies a physical storage page frame number. This will cause the Page Frame Structure corresponding to that frame to be displayed except for UVIRT storage. PFs corresponding to UVIRT storage are zeroed unless aliased by non-UVIRT storage. In the former case .MP will display then next non-UVIRT PF. In the latter it will display the aliasing non-UVIRT PF. See DP command for related information.

Results & Notes:

Page Frame Structures are allocated in contiguous storage from the address given by global variable: 

\texttt{_pft}

Output from the .MP command appears in one of three formats.

In-use Page Frame Structure
Idle Page Frame Structure.
Free Page Frame Structure.

For a description of the fields formatted by .MP select .MP Output Field Descriptions

For more examples using of the .M family of commands see: Exploring Memory Management.

Free Page Frame Structures

\begin{verbatim}
ffdf509c Free: BLink=0000f Flg=4 FLink=001da Blk=00001 Frame=0000d
ffdf50a8 Free: BLink=001f2 Flg=4 FLink=0003f Blk=00001 Frame=0000e
ffdf50b4 Free: BLink=001f1 Flg=4 FLink=001f1 Blk=00000 Frame=0000f
ffdf50c0 Free: BLink=001fe Flg=4 FLink=001f1 Blk=00000 Frame=00010
\end{verbatim}

Free Page Frame Structures.

Notes

- Free Page Frame Structures are chained in a double linked-list. The head of this list may be located as follows:
  1. Locate list structure whose address is given by \texttt{_pgFreeList}
  2. The first double-word of the list structure points to the pseudo-page frame structure that heads the free list.
  3. The second double-word contains the pseudo-frame number of the pseudo-PF. N.B. This marks the end of the linked list only.
  4. The backward pointer to the first true free PF is given by the 5 low order digits of the second double-word of the
pseudo-PF. This value may be used with the .MP command to format the first free PF provided it is not equal to the pseudo-frame number, which would imply no free PFs on the free list.

5. Subsequent free PFs are found by following the BLink until the pseudo-frame is encountered.
   • The Blk field has a residual field an is of no direct relevance to free Page Frame Structures.

For a description of the fields formatted by .MP select .MP Output Field Descriptions

--------------------------------------------

Idle Page Frame Structures

ffdfc8 Idle: pVP=ff1e6b9c Blink=01279 Flg=0 Flink=0004c Blk=0004a Frame=0127a
ffdfcdac Idle: pVP=ff1e6ba8 Blink=01261 Flg=0 Flink=0127a Blk=0004b Frame=01279
ffdfcc8c Idle: pVP=ff1e6c08 Blink=0125d Flg=0 Flink=01279 Blk=00066 Frame=01261

Notes

• Idle Page Frame Structures are chained in a double linked-list. The head of this list may be located as follows:

  1. Locate list structure whose address is given by __pgIdleList
  2. The first double-word of the list structure points to the pseudo-page frame structure that heads the idle list.
  3. The second double-word contains the pseudo-frame number of the pseudo-PF. N.B. This marks the end of the linked list only.
  4. The backward pointer to the first true idle PF is given by the 5 low order digits of the second double-word of the pseudo-PF. This value may be used with the .MP command to format the first idle PF provided it is not equal to the pseudo-frame number, which would imply no idle PFs on the idle list.
  5. Subsequent idle PFs are found by following the BLink until the pseudo-frame is encountered.

For a description of the fields formatted by .MP select .MP Output Field Descriptions

--------------------------------------------

In-use Page Frame Structures

ffdf5000 InUse: pVP=ff1d000 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00000
ffdf500c InUse: pVP=ff1f060 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001
ffdf5018 InUse: pVP=ff1f06c RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00002
ffdf5024 InUse: pVP=ff1f078 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00003

In-Use Page Frame Structures.

For a description of the fields formatted by .MP select .MP Output Field Descriptions

--------------------------------------------
.MP Output Field Descriptions

Output from .MP appears in one of is of the following forms:

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>pVP=</th>
<th>RefCnt</th>
<th>Flg=</th>
<th>ll=</th>
<th>sl=</th>
<th>Blk=</th>
<th>Frame=</th>
</tr>
</thead>
<tbody>
<tr>
<td>fddf500c InUse:</td>
<td>pVP=ff1df060 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fddfcdac Idle:</td>
<td>pVP=ff1e6ba8 Blink=01261 Flg=0 Flink=0127a Blk=0004b Frame=01279</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fddf50b4 Free:</td>
<td>BLink=001f1 Flg=4 FLink=0000d Blk=00001 Frame=0000f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each of the fields has the following meaning:

**address**

The linear address of the PF structure is given to the left of each display line.

**type**

The type of PF is displayed in the second column. Three types are possible: Free, Idle and InUse.

**pVP=**

The linear address of the associated Virtual Page Structure. See .MV Command for information on displaying Virtual Page Structures.

**RefCnt**

The number of PTEs that reference the frame of physical storage represented by this PF. A reference count greater than 1 indicates shared memory, some instances of which will be represented by VMCOs (see .MC command).

When a PTE is attached to an existing PF then the Refcnt is incremented.

When a page of memory is freed, the Refcnt is decremented. If it becomes zero the PF may be eligible for putting on the Idle list.

PFs corresponding to UVIRT storage are zeroed unless aliased by non-UVIRT storage. In either case no reference accounting is performed for UVIRT mappings.

**Blink=**

The backward or right link to the previous Idle or Free PF.

**Flink=**

The forward or left link to the next Idle or Free PF.

**Flg=**

PF flags.

The following flags are defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF_FAST</td>
<td>0x1</td>
<td>frame is fast memory</td>
</tr>
<tr>
<td>PF_BUSY</td>
<td>0x2</td>
<td>frame is busy</td>
</tr>
<tr>
<td>PF_FREE</td>
<td>0x4</td>
<td>frame is free</td>
</tr>
<tr>
<td>PF_RES</td>
<td>0x8</td>
<td>reserved</td>
</tr>
</tbody>
</table>

**Notes:** PF_FAST flag is set for some physical storage frames below 640K.

PF_BUSY signifies that access to the PF is being serialised by the page frame manager. This is normally followed by setting the VP_BUSY flag in the associated VP, if reset or setting the VP_WANTED flag and waiting on the the Block Id of the VP address. Under the debug kernel the thread slot of the VP semaphore owner is saved in vp_semowner (VP+0x0a) See .PB command for information on thread slots waiting on Block Ids.

**ll=**

Count of number of long-term lock requests active against this page frame. This is incremented when a request to
lock a range of pages of a memory object is made. It is also, but rarely, set to 1 to isolate page frames that have
caused trap 2 errors from which the system has recovered. See also .MO command output for flags relating to
memory object locking.

**sl=**
Count of number of short-term lock requests active against this page frame. This is incremented when a request to
lock a range of pages of a memory object is made. For related information, see .MO command output for flags
relating to memory object locking.

**Bik=**
Specifies the swap disk frame, loader block number or diagnostic flag depending on the flag settings of the
corresponding Virtual Page Structure pointed to by the pVP= field.

When **VP_DF** is set and **VP_DISCARDABLE** is reset then Bik= is the swap disk frame number that contains a copy
of the page frame.

When **VP_DISCARDABLE** is set and **VP_RESIDENT** is reset then the Bik= field is the Loader block Id. Except for a
special case noted below, this is a page index, starting from 1, into the objects of the module as an aggregated
whole, with the size of each object rounded up to a page boundary. The special case occurs when the memory object
that owes this page frame has an hnte set to system object id 0xffc0, *Discard Owner*. When this occurs the
following special block numbers may be used:

- 0x0ffe: System Infoseg
- 0x0ffd: Local Infoseg
- 0x0fffc: invalid LDT pages

When **VP_DF** and **VP_DISCARDABLE** are reset the Bik= usually indicates the last cross-linked swapper disk frame
(unless its zero), however under the debug kernel negative values are used to indicate errors or instances where
swapper frames have been freed because the corresponding PTE for the frame was found to be dirty. The following
error indicators are possible:

-1 When also Flg=9 then the physical frame caused a Trap 2 error, but the system
was able to recover the data. The frame is isolated from further use by setting ll=1,
refcnt=1 and PP_FREE flags are reset and pVP=pgVPBasePg
-3 A page-in operation failed with ERROR_SWAP_IO_PROBLEMS
-4 A page_out operation failed with PGPO_FAILED
-5 A page_out operation failed with ERROR_SWAP_FILE_FULL

Otherwise disk frame reclamation is indicated by Bik= values of: -1, -2, -7, -9 and 0xfff0.

For related **VP** information, see .MV command.

**Frame=**
This is the physical page frame number that this Page Frame Structure represents.

---

**.MV - Format Memory Virtual Page Structures (VPs)**

Display memory Virtual Page Structures (VPs).

**Syntax:**

```
.MV B F vpid L laddr Ln R
```

**Parameters:**

**B**
Display in-use (busy) Virtual Page Structures in sequential order.
**Note:** In-Use VPs are signified by a zero reference count and not by the [VP_BUSY](https://example.com) flag. See [Ref.](https://example.com) field in [MV](https://example.com) Output Field Description.

\begin{itemize}
  \item **F**  
  Display free Page Frame Structures.
  \item **L**  
  Follow left (forward) chain pointer. This is only of relevance to Free Virtual Page Structures since these are linked in a double linked chain. 
  \begin{itemize}
    \item **Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly and under some circumstances the debug kernel may hang. Use this option advisedly! From fix pack 29 for Warp V3 and base Warp V4 this has been fixed.
  \end{itemize}
  See [Free Virtual Page Structures](https://example.com) for information on locating Free VP chains.
  \item **R**  
  Follow right (backward) chain pointer. This is only of relevance to Free Virtual Page Structures since these are linked in a double linked chain. 
  \begin{itemize}
    \item **Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly and under some circumstances the debug kernel may hang. (.MVFL will cause this effect). Use this option advisedly! From fix pack 29 for Warp V3 and base Warp V4 this has been fixed.
  \end{itemize}
  See [Free Virtual Page Structures](https://example.com) for information on locating Free VP chains.
  \item **laddr**  
  Specifies the linear address of a specific Virtual Page Structure to be formatted. 
  \begin{itemize}
    \item An address expression may be specified.
  \end{itemize}
  \item **Ln**  
  Specifies the number of Virtual Page Structures to display from the starting criterion.
  \item **vpid**  
  Specifies a VP Id. This is an index in to the table of Virtual Page Structures, which are located in contiguous virtual storage.
\end{itemize}

**Results & Notes:**

Virtual Page Structures are allocated in contiguous storage from the address given by global variable: 

\_pgpVPBase

Output from the [MV](https://example.com) command appears in one of two formats.

\begin{itemize}
  \item In-use Virtual Page Structure
  \item Free Virtual Page Structure.
\end{itemize}

For a description of the fields formatted by [MV](https://example.com) select [MV Output Field Descriptions](https://example.com)

For more examples using of the [M](https://example.com) family of commands see: [Exploring Memory Management](https://example.com)

Free Virtual Page Structures

VPI=0d3e pVP=ff1e8ee8 free FLink=00000000 BLink=ff13280
VPI=0d3f pVP=ff1e8ef4 free FLink=ff1e9fec BLink=ff1e8cf0
VPI=0d40 pVP=ff1e8f00 free FLink=ff1e9fec BLink=ff1e8cf0
VPI=0d41 pVP=ff1e8f0c free FLink=00001000 BLink=02450030
VPI=0d42 pVP=ff1e8f18 free FLink=00000000 BLink=ff1e8f00
Free Virtual Page Structures.

Notes

• Free Page Frame Structures are grouped in bundles that are chained in a circular double link list. Each bundle comprises contiguous free VPs in the VP array. The chain pointers are only used by the head and tail of each bundle as follows:

  - For bundles of greater than one VP:
    1. **Blink** of the head points to the tail
    2. **Flink** of the head points to the head of the next bundle
    3. **Blink** of the tail points to the head of the previous bundle
    4. **Flink** of the tail is set to zero

  - For single VP bundles:
    1. **Blink** points to the head of the previous bundle
    2. **Flink** points to the head of the next bundle

The free VP chain is headed by a pseudo-VP whose **Blink** points to the head of the first true free bundle and whose **Flink** points to the last VP in the VP array. The pseudo-VP is located at global symbol:

```
_pgfVPHead
```

• Unless a free VP is the head or tail of a bundle then its **Flink** and **Blink** will retain values from its previous use. In particular it may be possible to glean information about a previous allocation as the **Flink** field overlays the Flg and Block fields and the **Blink** field overlays the HobPg and Hob fields of an In-use VP. In the example above VPI d41 was probably allocated to page 30 of hob 245. Using the following .MOC command might reveal who the owner was and who allocated this storage as follows:

```
.MOC 245
```

For a description of the fields formatted by .MV select .MV Output Field Descriptions

For more examples using of the .M family of commands see: Exploring Memory Management.

--------------------------------------------

In-use Virtual Page Structures

```
VPI=0000 pVP=ff1df000 Res Frame=0000 Flg=410 HobPg=0000 Hob=ff77 Ref=001 Own=000
VPI=0001 pVP=ff1df00c Res Block=0000 Flg=000 HobPg=0000 Hob=ff6c Ref=042 Own=000
VPI=0002 pVP=ff1df018 Res Frame=0bc5 Flg=410 HobPg=0000 Hob=ff6c Ref=001 Own=000
VPI=0003 pVP=ff1df024 Res Frame=0bc4 Flg=410 HobPg=027a Hob=0022 Ref=001 Own=000
```

Free Virtual Page Structures.

For a description of the fields formatted by .MV select .MV Output Field Descriptions

--------------------------------------------

.MV Output Field Descriptions

Output from .MV appears in one of is of the following forms:

```
VPI=0000 pVP=ff1df000 Res Frame=0000 Flg=410 HobPg=0000 Hob=ff77 Ref=001 Own=000
VPI=0001 pVP=ff1df00c Res Block=0000 Flg=000 HobPg=0000 Hob=ff6c Ref=042 Own=000
```
Each of the fields has the following meaning:

**VPI**
- The VP index into the array of VPs.

**pVP**
- The linear address of the VP.

**status**
- The status of the VP interpreted from the Flg field. The following values may appear:
  - SOW: Swap on Write flag (VP_SOW set)
  - Res: Page is resident (VP_RESIDENT set)
  - Dsc: Page is discardable (VP_DISCARDABLE set)
  - Swp: Page is swappable (VP_DISCARDABLE reset)
  - free: VP is free (vp_refcount=0)

**Block=nnnn**
- The cross-linked loader block number or swapper disk frame. This implies the virtual page is not attached to a PF. If it is:
  - discardable: then it is linked to a loader block id,
  - swappable: then it is linked to a swapper disk frame.

When the page is swappable (VP_DISCARDABLE reset) and does not have a disk frame (VP_DF reset) then the following special Block values may be used:

- 0: Allocate PF on demand
- 1: Allocate on demand zero-fill page
- 2: page is in a broken disk frame

**Frame=nnn**
- The virtual page is linked to PF nnnn. Refer to the .MP command for displaying information about the related page frame.

**Flink=**
- Forward link of a free VP. This is only of relevance to the VP at the head of a bundle of free VPs. See Free Virtual Page Structures for information on how free VPs are linked.

**Blink=**
- Backward link of a free VP. This is only of relevance to the VP at the head and tail of a bundle of free VPs. See Free Virtual Page Structures for information on how free VPs are linked.

**Flg=**
- VP flags.

The following flags are defined:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP_BUSY</td>
<td>0x001</td>
<td>page semaphore taken</td>
</tr>
<tr>
<td>VP_WANTED</td>
<td>0x002</td>
<td>page semaphore requested</td>
</tr>
<tr>
<td>VP_CACHE</td>
<td>0x004</td>
<td>search page cache for pf</td>
</tr>
<tr>
<td>VP_PFIDLE</td>
<td>0x008</td>
<td>cross linked to idle pf</td>
</tr>
<tr>
<td>VP_PF</td>
<td>0x010</td>
<td>cross linked to pf</td>
</tr>
<tr>
<td>VP_DF</td>
<td>0x020</td>
<td>has swap file disk frame</td>
</tr>
<tr>
<td>VP_DIRTY</td>
<td>0x040</td>
<td>contents written to - from pte</td>
</tr>
<tr>
<td>VP_SHDIRTY</td>
<td>0x080</td>
<td>shadow dirty bit (for VDMs)</td>
</tr>
<tr>
<td>VP_SOW</td>
<td>0x100</td>
<td>change to swappable on write</td>
</tr>
</tbody>
</table>
VP_PRIVATIZED 0x200 vp privatized
VP_RESIDENT 0x400 cannot be moved - value from pte
VP_DISCARDABLE 0x800 1 = discardable, 0 = swappable

Notes:

PF_BUSY signifies that access to the VP is being serialised by the page frame manager.

VP_WANTED signifies that a thread is waiting to mark the VP busy. The thread will wait on a Block id of the VP address. Under the debug kernel the thread slot of the VP semaphore owner is saved in vp_semowner (VP+0x0a) see Own= field of the .MV display. See .PB command for information on thread slots waiting on Block Ids.

HobPg=
The relative page number of the memory object that this VP is assigned to. See Hob= field below.

Hob=
The hob of the memory object to which this page is assigned.

Note: Use .MOC hob

to obtain the virtual address and owner information relating to this VP. See .MO command for more information.

Ref=
The number of memory objects sharing this page of data. A reference count greater than 1 indicates shared memory, some instances of which will be represented by VMCOs (see .MC command) and others by aliases (see .ML command).

The reference count is incremented and decremented according to usage. When the count becomes zero the VP is no longer in use and any committed physical storage or swapper storage may become eligible for freeing.

UVIRT storage is not represented by VPs thus reference accounting is not performed.

Own=
The thread slot number of the current owner of the VP semaphore. This field is only used in the debug kernel and will only have significance if the VP_BUSY or WP_WANTED flags are set. See .PB command for information on thread slots waiting on Block Ids.

.N - Display Dump Information Summary

Display information saved by from the operating system when the stand alone dump procedure was initiated.

Syntax:

.N

Parameters:

none

Results & Notes:
.N command displays information saved when the kernel routine RASRST is entered at sand-alone dump initiation.

.N displays the following information:

gdtr_lim: 1FFF
gdtr_base: 7C3E5000
idtr_lim: 03FF
idtr_base: FFE00150
ldtr_reg: 0028
lo_data_sel: 0400
hi_data_sel: 0400
trace_buf_addr: 0B490400
sys_anchor_sel: 0070
arena_base: FEB1F020
max_threads: 0101
phys_page_dir: 00D60000
vm_object_ptr: FEC80020
StartInit_Data: 00000140
dcm_ote_start: FFF0A92F
CurProcPid: 000D
TaskData: 095C0400
FirstPacket: 158A
LastPacket: 04C0
SysSemDataTable: 53A60400
GDT_Buffers: 00A80138
PapTCBPtrs: 0B6B0400
callerSS: 00E8
callerESP: 00000FCC
savePage: 00241467

Each of the items displayed has the following significance:

- **gdtr_lim**: The current GDTR register limit value.
- **gdtr_base**: The current GDTR register base address.
- **idtr_lim**: The current LDTR selector limit.
- **idtr_base**: The current LDTR selector base address.
- **ldtr_reg**: The current LDT selector.
- **lo_data_sel**: Selector for DOSGROUP segment.
- **hi_data_sel**: Selector for DOSHIGHDATA segment.
- **trace_buf_addr**: Offset:selector address of ras_stda_addr, the selector for the system trace buffer.
- **sys_anchor_sel**: Selector for the SAS.
- **arena_base**: Value of _parvmOne, the pointer to the first VMAR.
- **max_threads**: Maximum Thread Slot Number.
- **phys_page_dir**: Value of cr3 register (that is, the physical address of the page directory table).
- **vm_object_ptr**: Value of _pobvmOne, the pointer to the first VMOB.
StartInit_data

Value of _StartINITData.

dcm_ote_start

Address of DOSCALLS.DLL OTE

CurrProcId

Current process ID.

TaskData

Offset:selector address of scheduler global data.

FirstPacket

First word of the first device driver strategy 2 request packet in packet pool.

LastPacket

First word of the last device driver strategy 2 request packet in packet pool.

SysSemDataTable

Offset:selector address of SysSemDataTable.

GDT_Buffers

GDT selector for buffer segment. The low order word of this field should be ignored.

PapTCBPtr

Offset:selector address of papTCBPtr. The word value at this label is an offset from the DOSGROUP selector (400) to the thread slot table.

callerSS

The SS selector on entry to RASRST, the stand alone system dump entry point within the kernel.

Note: If this dump was taken in interrupt mode the SS selector will be E8. Further more, if that last device driver to use the interrupt stack is KDB$ then it is possible the the dump process was initiated with the Ctrl-Alt-Numlock-Numlock sequence. If the dump was taken in kernel mode then the SS selector will probably be 30 and the dump will have been initiated because of a trap or call to DosForceSystemDump API.

callerESS

The ES register on entry tp RASRST, the stand alone system dump entry point within the kernel.

savePage

Page directory entry 0. This data is overwritten by the dump process.

.O - Override default behaviour

Override the default system behaviour for handling certain events.

Syntax:

.O  Q
N  n

Parameters:

Q
Query override settings in effect.

N
Activate (turn on) overrides for event n.
Deactivate (turn off) overrides for event \( n \).

\( n \)

Specifies the override event number. Currently only one override event is defined, which is event number 1. When active this forces breakpoints to be serviced by the Kernel Debugger instead of the **DosDebug** or **DosPTrace** APIs. This only affects debugging situations where a debugging application is active under the Kernel Debugger.

**Results & Notes:**

This command was introduced with fix pack 29 for Warp 3.0 and fix pack 1 for Warp 4.0.

Prior to the introduction of this command, the only way to override **DosDebug** and **DosPTrace** was to zero to address of the Debug Control Block ((ptda_pdcb +0x38)) in the PTDA of the debugee process.

**.OQ** displays information about overrides in a tabular form as shown in the following example:

```
#.OF 1
DosDebugAPI Override is now OFF.
#.OQ
Override(s) Query Information
# | Override          |   Status
--------------------------------------------
1 | DosDebugAPI      |   OVERRIDE OFF
--------------------------------------------
#.ON 1
DosDebugAPI Override is now ON.
#.OQ
Override(s) Query Information
# | Override          |   Status
--------------------------------------------
1 | DosDebugAPI      |   OVERRIDE ON
--------------------------------------------
```

**.P - Display Process Status**

Display process and thread status information from the Per Task Data Area (PTDA), Thread Control Block (TCB) and Thread Swappable Data (TSD).

**Syntax:**

```
.P                                                         `
#                                                             `
*. slot
```

**Parameters:**

slot

Display process status for thread slot `slot`.

The following short-hand may be used for the slot number:

* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

```
_TaskNumber
```
The debugger default thread slot. This defaults to the current slot unless overridden by the .S command.

If no slot number is given then all thread slots are displayed, grouped by process.

Results & Notes:

The .P command locates a thread's TCB from either the thread slot table, the linear address of which is given by global variable: _papTCBSlots

or by traversing the process tree using TCBpTCBNext (TCB +0x14), TCBpPTDA (TCB +0x08) and ptda_pTCBHead (PTDA + 0x20) fields. Output from the .P command appears in tabular form as follows:

<table>
<thead>
<tr>
<th>Slot</th>
<th>Pid</th>
<th>Ppid</th>
<th>Csid</th>
<th>Ord</th>
<th>Sta</th>
<th>Pri</th>
<th>pTSD</th>
<th>pPTDA</th>
<th>pTCB</th>
<th>Disp</th>
<th>SG</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0001</td>
<td>blk</td>
<td>0100</td>
<td>ffe3a000 ffe3c7d4 ffe3c61c 1eb4</td>
<td>00 *ager</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0002</td>
<td>blk</td>
<td>0200</td>
<td>7b7ca000 ffe3c7d4 7b9c8020 1f3c</td>
<td>00 *tstd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0003</td>
<td>blk</td>
<td>0200</td>
<td>7b7cc000 ffe3c7d4 7b9c81d8 1f50</td>
<td>00 *ctxh</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0004</td>
<td>blk</td>
<td>081e</td>
<td>7b7ce000 ffe3c7d4 7b9c8390 1f48</td>
<td>00 *kdb</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0005</td>
<td>blk</td>
<td>0800</td>
<td>7b7d0000 ffe3c7d4 7b9c8548 1f20</td>
<td>00 *lazyw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>0001</td>
<td>0000</td>
<td>0000</td>
<td>0006</td>
<td>blk</td>
<td>0800</td>
<td>7b7d2000 ffe3c7d4 7b9c8700 1f3c</td>
<td>00 *asynchr</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*0008</td>
<td>0002</td>
<td>0001</td>
<td>0002</td>
<td>0001</td>
<td>blk</td>
<td>0500</td>
<td>7b7d6000 7b9e4020 7b9c8a70 1eb8</td>
<td>01 pmshell</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000a#</td>
<td>0002</td>
<td>0001</td>
<td>0002</td>
<td>0002</td>
<td>blk</td>
<td>0800</td>
<td>7b7da000 7b9e4020 7b9c8de0 1ed4</td>
<td>01 pmshell</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each of the fields has the following meaning:

**Slot**
The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

* to the left to signify the current (or last) dispatched thread.
# to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

Slot may be found in the TCBNumber (TCB + 0x2) field of the TCB

**Pid**
The process id this thread slot is assigned to.

**Ppid**
The parent process id that created this thread. A value of zero signifies a detached process.

**Csid**
The command subtree id.

This is normally the same value as the Pid. When the parent process dies any orphaned children are adopted by their grandparent by making Ppid equal to the grandparent's Pid. Each orphan inherits the Csid of its dying parent. This mechanism ensures that orphaned PTDA's are not retained for returning termination information to their parent (via DosWaitChild).

Csid is taken from the Csid (PTDA +0x4be (H/R: +0x4b6)) field of the PTDA

**Ord**
The thread ordinal for this thread slot. This is the unique thread id assigned to the thread within the process to which it belongs.

Ord is taken from the TCBOrdinal (TCB+ 0x0) field of the TCB

**Sta**
The thread's ascending scheduler state taken from the TCBState (TCB +0x161) field.

Except when a state transition is progress this is the same as the current state of the thread (see Qst field of the .PQ command.)

The following states are possible:

<table>
<thead>
<tr>
<th>abrv</th>
<th>Sta</th>
<th>TCBState description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>value</td>
</tr>
</tbody>
</table>
--- STATE_VOID 0 Uninitialized or Dead thread
rdy STATE_READY 1 Thread ready to run
blk STATE_BLOCKED 2 Blocked on a Block Id
sus STATE_SUSPENDED 3 *** Not in Use ***
crt STATE_CRITSEC 4 Blocked by another CritSec thread (after attempting to run)
run STATE_RUNNING 5 Thread currently running
bst STATE_READYBOOST 6 Ready, but apply an IO boost after swapping in a TSD
tsd STATE_TSD 7 Thread waiting for the TSD daemon to page in the TSD.
dly STATE_DELAYED 8 Delayed TKWakeup (Almost Ready)
frz STATE_FROZEN 9 Frozen Thread via DosSuspendThread, DosCreateThread, DosExecPgm or DosSystemService
gsk STATE_GETSTACK 10 TSD daemon is waiting for the page manager to page in a TSD
bad STATE_BADSTACK 11 TSD failed to page in

Notes:

The scheduler manages threads on queues by priority and state. See .PQ command for displaying scheduler queues.

The scheduler uses a finite state machine to manipulate thread queues. TCBQState and TCBState are the state transition drivers. They hold a thread's current and desired state. Except during a state transition current and desired state will be identical.

STATE_RUNNING is set when the next potential runner has been selected. The running thread's context is then switched and various dispatcher flags checked before finally giving control to user code. It is therefore possible for the running state to be set and for the user code not to run.

STATE_READYBOOST is a modified ready state and never becomes the current state, instead a priority boost is applied and the state becomes STATE_READY

STATE_CRITSEC This state applies to non-critical section threads only. It is only set when a critical section thread within the same process has given up the processor, while still in critical section, and another thread in the same process is selected to run. If this thread is thread 1 of the process and there are pending signals to process, the thread's signal handler will be dispatched. When there are no more pending signals or this thread is not thread 1 then STATE_CRITSEC will be set.

STATE_FROZEN is normally only seen when an application uses the DosSuspendThread API or creates a thread (or process) that is initially suspended. DosSystemService is used by the session manage to freeze all threads of a process in one system call.

Many states are transient accordingly the persistent appearance of a particular state might indicate a problem of the following nature:

rdy Many ready threads might indicate contention for processor time. Tends to indicate the existence of a higher priority CPU bound thread.

run Under the Dump Formatter this would indicate a trapped or processor bound thread. Under the Kernel Debugger a processor bound thread.

blk All threads blocked could indicate no-work or a
deadlock. Under Dump Formatter this would imply a manually invoked dump using Ctrl-Alt-Numlock-Numlock or use of the DosForceSystemDump API.

---

The void state is rarely seen. Under Dump Formatter this probably indicates an incorrect version of the Dump Formatter for the system dumped.

crt
Another thread in the same process is in critical section and is either blocking without exiting critical section or is processor bound.

dly
Another thread is processor bound.

frz
A deadlock, loop or no-work for the freezing thread

sus
Is not used, so probably indicates a mismatch between the Dump Formatter and dump.

tsdf
Physical storage overcommitted. Swapper very large. System may be thrashing.

bst
Physical storage overcommitted. Swapper very large. Very busy processor bound system. System may be thrashing.

gsk
Physical storage overcommitted. Swapper very large. Very busy processor bound system. System may be thrashing.

bad
Excessive swapper. System may die. Physical storage overcommitted. Very busy processor bound system.

---

Pri
Thread priority (word length field) in TCBPriority (TCB +0x168).

This is the current priority calculated by the scheduler based upon priority class (TCBPriClass (TCB +0x164)), priority class level (TCBPriLevel (TCB +0x165)) and priority boosts (TCBPriClassMod (TCB +0x166)).

The following priority classes are defined:

<table>
<thead>
<tr>
<th>class</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS_IDLE_TIME</td>
<td>0x01</td>
<td>Idle-Time class</td>
</tr>
<tr>
<td>CLASS_REGULAR</td>
<td>0x02</td>
<td>Regular class</td>
</tr>
<tr>
<td>CLASS_TIME_CRITICAL</td>
<td>0x03</td>
<td>Time-Critical class</td>
</tr>
<tr>
<td>CLASS_SERVER</td>
<td>0x04</td>
<td>Client-Server Server class</td>
</tr>
</tbody>
</table>

The following priority boosts (class modifiers) are defined:

<table>
<thead>
<tr>
<th>Boost</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSMOD_KEYBOARD</td>
<td>0x04</td>
<td>Keyboard boost</td>
</tr>
<tr>
<td>CLASSMOD_STARVED</td>
<td>0x08</td>
<td>Starvation boost</td>
</tr>
<tr>
<td>CLASSMOD_DEVICE</td>
<td>0x10</td>
<td>Device I/O Done Boost</td>
</tr>
<tr>
<td>CLASSMOD_FOREGROUND</td>
<td>0x20</td>
<td>Foreground boost</td>
</tr>
<tr>
<td>CLASSMOD_WINDOW</td>
<td>0x40</td>
<td>Window Boost</td>
</tr>
</tbody>
</table>
Note:

**CLASSMOD_KEYBOARD** has no effect on **CLASS_SERVER**

The priority level is a value between 0x0 and 0x1f.

Priority class and modifier values are logically ORed to form an index into the priority class translation table, which is located at global symbol:

```
_schPriClassTbl
```

The resulting value is logically ORed with the priority level. The final value is subject to the minimum thread priority (TCBPriMin [TCB +016a]).

Priority boosts do not affect the priority of idle and time-critical threads.

Priority level has little or no effect on the priority of boosted regular and server class threads.

**pTSD**

Linear address of the TSD control block associated with this thread this thread taken from the TCBpTSD [TCB +0x0c].

Note:

The TSD contains the ring 0 stack for the associated thread. For the current thread this is addressable from selector 30 however the base address of selector 30 is entirely different from TCBpTSD. This is because the two addresses are aliased using two PTEs to pin the same physical frame. This device allows the TSD for be accessed out-of-context by the system, at the same time protecting system code from erroneous stack references.

**pPTDA**

Linear address of the control block representing the process to which this thread belongs. The address is taken from TCBpPTDA [TCB +0x08].

**pTCB**

Linear address of the TCB control block which represents the thread.

Note:

The output from .P is ordered by process and child process. TCBs are initially located from the thread table then the chain pointer TCBpTCBNext [TCB +0x14] is used to locate the remaining threads of a process.

Under the Dump Formatter .P will occasionally miss a thread because of the non-sequential manner in which the thread table slots are re-used. To ensure all active threads are displayed use .PL, or .PQ commands.

**Disp**

The displacement into the TSD for the current thread that the dispatcher will use for its ESP after having switched back to this thread's context.

This value is calculated from TSDKernelESP, therefore requires the TSD to be present. If the TSD is not present then a blank value is given. The TSD may be forced present under the Kernel Debugger by use of the .I command.

**SG**

Screen Group ID currently assigned to this process.

The Screen Groups ID is taken from the console locus structure (Cons_Loc +0x2) embedded in the PTDA ([PTDA +0x526 (H/R: +0x51e)]).

**Name**

The name of the primary executable running in this process.

Except for process 1 and Dos Virtual Machines the name is obtained from the hmte stored in ptda_module [PTDA +0x5a6 (H/R: +0x59e)]. If the SMTE is paged in then the name is taken from the file name pointed to by smte_path otherwise it it taken from mte_module and prefixed with an ! point. See .LM command for information on formatting loader control blocks.
Process 1 comprises internal threads, that is threads which run in the kernel and are not separately loaded modules. _ptda_module_ is zero for this process so the Dump Formatter and Kernel Debugger translate the Tids for Pid 1 as follows:

<table>
<thead>
<tr>
<th>Tid</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*ager</td>
<td>Ager thread used for compressing the Swap File.</td>
</tr>
<tr>
<td>2</td>
<td>*tsd</td>
<td>Scheduler's Daemon Thread used to page in TSDs</td>
</tr>
<tr>
<td>3</td>
<td>*ctxh</td>
<td>Default Global Context Hook dispatching thread.</td>
</tr>
<tr>
<td>4</td>
<td>*kdb</td>
<td>Kernel Debugger Daemon thread used to process page-in requests from the .ID command</td>
</tr>
<tr>
<td>5</td>
<td>*lazyw</td>
<td>File system cache lazy writer thread.</td>
</tr>
<tr>
<td>6</td>
<td>*asyncr</td>
<td>File system asynchronous read ahead thread.</td>
</tr>
<tr>
<td>7</td>
<td>*sysinit</td>
<td>System initialisation thread.</td>
</tr>
<tr>
<td>8-n</td>
<td>Other transient internal threads associated with system initialisation have a blank name.</td>
<td></td>
</tr>
</tbody>
</table>

Virtual DOS Machines run the DEM component of OS/2 to provide DOS emulation. DOS programs are loaded by the DEM and not known to the (OS/2) loader. Thus _ptda_module_ is zero and the Kernel Debugger and Dump Formatter use the name *vdm* to indicate a VDM. The PSP of the first loaded DOS program in a process may be located from CurrentPDB (PTDA +0x2ea), which contains its segment address. The preceding paragraph contains the DOS arena record, the last 8 bytes of which contains the DOS program name currently executing.

--------------------------------------------

Scheduler Finite State Machine
.PB - Display Blocked Thread Information

Display information about all blocked threads.

Syntax:

```
.PB
```

Parameters:

```
slot
```

Display user information for thread slot `slot`.

The following short-hand may be used for the slot number:

```
* 
```

The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

```
_TaskNumber
```
The debugger default thread slot. This defaults to the current slot unless overridden by the .S command.

If no slot number is given then all thread slots are displayed in slot number order.

**Results & Notes:**

The .PB command locates each thread’s TCB from the thread slot table, the linear address of which is given by global variable: __papTCBSlots

or by traversing the process tree using TCBpTCBNext (TCB +0x14), TCBpPTDA (TCB +0x08) and ptda_pTCBHead (PTDA + 0x20) fields.

Output is displayed only if a thread is blocked on a Block-Id. It appears in tabular form as follows:

<table>
<thead>
<tr>
<th>Slot</th>
<th>Sta</th>
<th>BlockID</th>
<th>Name</th>
<th>Type</th>
<th>Addr</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>blk</td>
<td>fff11050</td>
<td>*ager</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>blk</td>
<td>fff74f59</td>
<td>*tsd</td>
<td>_tkTSDDaemon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>blk</td>
<td>fff43c78</td>
<td>*ctxh</td>
<td>_kmCTXHDaemon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>blk</td>
<td>fff7545a</td>
<td>*kdb</td>
<td>_tkKDBDaemon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>blk</td>
<td>fff02dfc</td>
<td>*lazyw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>blk</td>
<td>fff111d4</td>
<td>*asyncr</td>
<td>_AsyncReadSem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>blk</td>
<td>fffe000e</td>
<td>pmshell</td>
<td>RamSem</td>
<td>074b:06d6</td>
<td></td>
</tr>
<tr>
<td>000a</td>
<td>blk</td>
<td>ffca0002</td>
<td>pmshell</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000b</td>
<td>blk</td>
<td>fffd000b</td>
<td>pmshell</td>
<td>MuxWait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000c</td>
<td>blk</td>
<td>fffd000c</td>
<td>pmshell</td>
<td>MuxWait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000d</td>
<td>blk</td>
<td>04000df0</td>
<td>pmshell</td>
<td>DosSem</td>
<td>0400:0df0</td>
<td>CtrlNumLkQ</td>
</tr>
<tr>
<td>0007</td>
<td>blk</td>
<td>fe750a10</td>
<td>pmshell</td>
<td>Sem32</td>
<td>8001 0019</td>
<td>vhevLazyWrite</td>
</tr>
<tr>
<td>0010</td>
<td>blk</td>
<td>fe728dcc</td>
<td>pmshell</td>
<td>Sem32</td>
<td>8001 0001</td>
<td>8rvReq</td>
</tr>
<tr>
<td>0011</td>
<td>blk</td>
<td>fffe0006</td>
<td>pmshell</td>
<td>RamSem</td>
<td>d0c7:0020</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>blk</td>
<td>fffd0012</td>
<td>pmshell</td>
<td>MuxWait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*0013</td>
<td>blk</td>
<td>fffe0007</td>
<td>RamSem</td>
<td>d09f:0bc0</td>
<td>memory_pool + 127</td>
<td></td>
</tr>
<tr>
<td>0014#</td>
<td>blk</td>
<td>fffe0008</td>
<td>RamSem</td>
<td>d09f:0bc8</td>
<td>memory_pool + 12f</td>
<td></td>
</tr>
</tbody>
</table>

Each of the fields has the following meaning:

**slot**

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

* to the left to signify the last dispatched thread.

# to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

**Slot** may be found in the TCBNumber (TCB + 0x2) field of the TCB

**Sta**

The ascending or desired state of the thread. This should always appear as blk for the .PB command, however Dump Formatter does not check the thread state so formats all threads. Those whose state is not blk should be ignored. From fix pack 29 for Warp V3 and base Warp V4 this has been fixed so that only blocked threads are formatted. See Scheduler Finite State Machine and .PQ command for more information on thread states.

**BlockID**

The token used by TKSleep and TKWapeUp to sleep and wake a thread on an event.

The BlockId is taken from TCBSleepID (TCB + 0x18c).

The BlockId is a conventional value. A number of conventions are used by various system components. Usually the BlockId is constructed so to be unique across all conventions. Frequently it will refer to the address of an associated resource, such as a system control block, or a field within a control block. See discussion of the Type field below for more information on interpreting BlockIds.

**Name**

The name of the primary executable running in this process.

See name field description of the .P command for further information.

**Type**
Interpretation of the use of the **BlockID** in conjunction with double word **TCB_SemInfo** (TCB + 0x14c) and double word **TCB_SemDebugAddr** (TCB + 0x150).

The following **Types** are recognised by the Dump Formatter and Kernel Debugger:

**RamSem**

The thread is waiting on a **RamSem** or **FastSafeRamSem**.

The high word of the **BlockID** is **0xfffe**, the low word is the **RamSemID** taken from the **RamSem** structure.

The **Addr** field is taken from **TCB_SemInfo**. This is a **selector:offset** address of the RAMSEM. The **RamSem** may be imbedded within a **FastSafeRamSem** or a **PMFastSafeRamSem**.

The **Symbol** displayed is either that of the **TCB_SemDebugAddr** or if -1, the **RamSem** address. See LN command for information on displaying symbols.

**MuxWait**

The thread is waiting on multiple events.

The high word of the **BlockID** is **0xfffd**, the low word is the **slot** of the waiting thread.

**TCB_SemInfo** and **TCB_SemDebugAddr** are not used with a **MuxWait BlockID**.

To locate the semaphores that comprise a given **MuxWait** proceed as follows:

1. Locate the **MuxWait** table at symbol **MuxTable**. Display this using DB for convenience. This table comprises 9 byte entries whose format is given by the **MuxTableEntry** structure.
2. Scan the **MuxTable** for entries that have this thread's slot number (+0x2 into each entry).
3. Of those entries, select those with non-zero **MuxType** (+4 into each entry).
4. Choose one of the following:
   - **SysSem**
     - the last double word is the linear address of a system semaphore structure. Use the technique described below under **SysSem** for interpreting the **SysSem**.
   - **RamSem**
     - the last double word of the entry contains the **RamSem** handle, the high word is the **hob** of the memory object containing the **RamSem**. The low word is the offset into the object where the **RamSem** is located. Use the technique described above under **RamSem** for interpreting the **RamSem**.
   - **Physical RamSem**
     - the last double word is the physical address of the **RamSem**.
   - **32-bit event sem**
     - the last double word is the physical address of a 32-bit event semaphore. See **Sem32** below.

See **Mux Semaphore** example debugging log for an explicit example of using this technique.

**Addr** and **Symbol** fields are blank.
**ReqPkt**

The thread is waiting for an I/O request packed to complete.

The **BlockID** is the **Selector:Offset** address of the request packet. The Selector is the **DOSGROUP** kernel selector and should be selector 400.

The address should lie between addresses at global symbols: **FirstPacket** and **LastPacket**.

See the Physical Device Driver Reference manual for information on device driver request packets.

**Addr** and **Symbol** fields are blank.

**SysSem**

The thread is waiting on a system semaphore.

The **BlockID** is the **Selector:Offset** address of the **SysSemTblStruc** structure. The Selector is the **DOSGROUP** kernel selector and should be selector 400.

The address should lie within the System Semaphore Data Table, located at symbol **SysSemDataTable** for length 256*6 bytes.

Offset +0 of each table entry contains the owner’s thread number.

The name associated with the semaphore may be located as follows:

1. Locate the **SysSem RMP** segment by displaying double word at symbol **SysSemRMPHdl**. The high word is the selector for the semaphore RMP.
2. Display the System Semaphore RMP using **DB**. The first 0x14 bytes is the RMP header. The remainder comprises variable length records. The first word of each record is its length and therefore the relative offset to the beginning of the following record. Offset 2 of each record is the semaphore data table selector offset.
3. Scan the RMP looking for an offset that matches the low word of the **BlockID**. When found the remaining bytes of the RMP record is the semaphore name (with the top two bytes overlaid by the semaphore offset).

See **System Semaphore** example debugging log for an explicit example of using this technique.

**Addr** and **Symbol** fields are blank.

**DosSem**

The thread is waiting on an internal **RamSem** or blocking on an address in the kernel’s **DOSGROUP** segment.

The **BlockID** is the **selector:offset** of the **DosSem**. The Selector is the **DOSGROUP** kernel selector and should be selector 400. The offset does not lie in the System Semaphore Data Table or the I/O Request Packet Table.

**Addr** is the **BlockID** formatted as **selector:offset**.

The **Symbol** displayed is either that of the **TCB_SemDebugAddr** or if -1, the **DosSem** address. See **LN command** for information on displaying symbols.

**Sem32**

The thread is waiting on a 32-bit semaphore.

The **BlockID** is the address of the **32-bit Semaphore structure**.

**TCB_SemInfo** contains the semaphore handle. This is of the form:

8001nnnn

Shared 32-bit semaphore. **nnnn** is the (double-word) index into the shared semaphore table located at symbol **_pShSemTbl**. Each entry is an address of the corresponding 32-bit
Private 32-bit semaphore. \textit{nnnn} is the (double-word) index into the private semaphore table located at \texttt{pPrSemTbl (PTDA +0x4cc (H/R: +0x4c4))}. Each entry is an address of the corresponding 32-bit semaphore structure. (that is, the \texttt{Sem32 BlockID}).

\textbf{Addr} field is the semaphore handle formatted as two words.

The \textit{Symbol} displayed is either that of the \texttt{TCB_SemDebugAddr} or if -1, the \texttt{Sem32 address}. See \texttt{LN command} for information on displaying symbols.

Use the \texttt{.D SEM32 command} with the \textit{BlockID} to format the 32-bit semaphore.

**Buffer**

The thread is waiting for a file system buffer.

The \textit{BlockId} is the \textit{selector:offset} address of the buffer. The high word is the buffer selector and should be \texttt{a8}.

The \textit{Addr} and \textit{Symbol} fields are blank.

**SFT**

The thread is waiting for a SFT entry.

The \textit{BlockId} is the \textit{selector:offset} address of the SFT. The high word is the buffer selector and should be one that is listed in the SFT table pointed to by \texttt{c0:0}.

The \textit{Addr} and \textit{Symbol} fields are blank.

**ChildWait**

The thread is waiting in DosWaitChild for a child process to terminate.

The high word of the \textit{BlockID} is the \texttt{ptda_Pid offset} from selector 30 (0xffca).

The low word of the \textit{BlockID} is the \textit{Pid} to which this thread belongs.

**blank type**

The thread is waiting on a \textit{BlockId} that the Kernel Debugger and Dump Formatter have not been able to identify.

\textit{Addr} field is blank.

The \textit{Symbol} displayed is either that of the \texttt{TCB_SemDebugAddr} or if -1, the \texttt{Sem32 address}. See \texttt{LN command} for information on displaying symbols.

\textbf{Notes}:

The \textit{BlockId} interpretation is not exact. A device driver, for example, could call \texttt{DevHlp_ProcBlock} using a value for \texttt{BlockID} that conflicts with another convention.

Under the Debug kernel only, \texttt{TCB_SemDebugAddr} is used to record the creator's address of kernel, system and RAM semaphores. If it is not used it is set to 0xffffffff.

\textit{ChildWait BlockIDs} might be missed by the Dump Formatter and Kernel Debugger. Look out for \textit{BlockIDs} of the form \texttt{0xffca????}. \textit{ChildWait} is correctly reported from fix pack 29 for Warp V3 and base Warp V4.

Some \texttt{Sem32 BlockIDs} are missed by the Dump Formatter. Check \texttt{TCB_SemInfo} for a 32-bit semaphore handle and \textit{BlockIds} of the form \texttt{0xfe????}.

If \textit{BlockId} is a linear address owned by \texttt{ksem} then the semaphore is a \texttt{Kernel Semaphore}. However, not every \texttt{KSEM} is owned \texttt{ksem} owned memory. Under the ALLSTRICT kernel, a \texttt{KSEM} may be readily identified from the first 4 bytes, which have the signature "KSEM". Under the \texttt{RETAIL} and \texttt{HSTRICT} kernels, the \textit{Blockid} is chosen to be the address (or handle) of the \texttt{KSEM}. Under the ALLSTRICT kernel, Event \texttt{KSEMs} use the \texttt{KSEM handle+4 as the BlockId}. Use \texttt{.D KSEM command} with the \textit{BlockID} to format a \texttt{KSEM}.

In general a \textit{BlockID} will be chosen to be meaningful to the programs using it. Often it is an address of a resource that needs to be serialised. Where no other information is given one should try:
to try to establish an owner of the resource represented by the BlockID

to try to establish a meaningful symbol associated with the BlockID

to try to establish the API or call the lead to the thread waiting on the BlockID (see .K command)

Unwind the User's Stack

Addr
The address of the semaphore associated with this BlockID
See Type field discussion above for more precise information.

Symbol
Either the symbolic address of the creator or of the associated semaphore.
See Type field discussion above for more precise information.

.PQ - Display Scheduler Queue Information

Display scheduler thread queue information for all (active) threads.

Syntax:
.PRO

Parameters:
.slot
Display queue status for thread slot slot.
The following short-hand may be used for the slot number:

* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:
  _.TaskNumber

# The debugger default thread slot. This defaults to the current slot unless overridden by the .S command.

If no slot number is given then all thread slots are displayed in slot number order.

Results & Notes:
The .PQ command locates each thread's TCB from the thread slot table, the linear address of which is given by global variable:
or by traversing the process tree using TCBpTCBNext (TCB +0x14), TCBpPTDA (TCB +0x08) and ptda_pTCBHead (PTDA + 0x20) fields.

Output from the .PQ command appears in tabular form as follows:

<table>
<thead>
<tr>
<th>Slot</th>
<th>QSt</th>
<th>Pri</th>
<th>pTCB</th>
<th>PriNextQ</th>
<th>PriPrevQ</th>
<th>PriHigh</th>
<th>PriLow</th>
<th>PriNext</th>
<th>PriPrev</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>blk</td>
<td>0100</td>
<td>ffe3c61c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>blk</td>
<td>0200</td>
<td>7b9c8020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>blk</td>
<td>0200</td>
<td>7b9c81d8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>blk</td>
<td>081f</td>
<td>7b9c8390</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>blk</td>
<td>0800</td>
<td>7b9c8458</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>blk</td>
<td>0800</td>
<td>7b9c8700</td>
<td>7b9cb3b0</td>
<td>7b9c9830</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>blk</td>
<td>0500</td>
<td>7b9c8a70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*0008</td>
<td>blk</td>
<td>0800</td>
<td>7b9c8de0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>blk</td>
<td>0800</td>
<td>7b9c8f90</td>
<td>7b9ca960</td>
<td>7b9ca960</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000a</td>
<td>blk</td>
<td>0800</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000b</td>
<td>blk</td>
<td>0800</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000c</td>
<td>blk</td>
<td>0800</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td>7b9c9150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each of the fields has the following meaning:

**slot**

The unique (hexadecimal) index in to the thread slot table of all threads. This value may be flagged with:

* to the left to signify the current (or last) dispatched thread.
# to the right to signify the Kernel Debugger default thread slot.

**QSt**

The thread’s descending or current scheduler state taken from the TCBQState (TCB +0x160) field. Except when a state transition is progress this is the same as the desired state of the thread (see Sta field of the .P command.)

The following states are possible:

<table>
<thead>
<tr>
<th>Abbr</th>
<th>QSt</th>
<th>TCBQState Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>STATE_VOID</td>
<td>0</td>
<td>Uninitialized or Dead thread</td>
</tr>
<tr>
<td>rdy</td>
<td>STATE_READY</td>
<td>1</td>
<td>Thread ready to run</td>
</tr>
<tr>
<td>blk</td>
<td>STATE_BLOCKED</td>
<td>2</td>
<td>Blocked on a Block Id</td>
</tr>
<tr>
<td>sus</td>
<td>STATE_SUSPENDED</td>
<td>3</td>
<td>*** Not in Use ***</td>
</tr>
<tr>
<td>crt</td>
<td>STATE_CRITSEC</td>
<td>4</td>
<td>Blocked by another CritSec thread (after attempting to run)</td>
</tr>
<tr>
<td>run</td>
<td>STATE_RUNNING</td>
<td>5</td>
<td>Thread currently running</td>
</tr>
<tr>
<td>tsd</td>
<td>STATE_TSD</td>
<td>7</td>
<td>Thread waiting for the TSD daemon to page in the TSD.</td>
</tr>
<tr>
<td>bst</td>
<td>STATE_READYBOOST</td>
<td>6</td>
<td>current state never set to this value – see note below.</td>
</tr>
<tr>
<td>dly</td>
<td>STATE_DELAYED</td>
<td>8</td>
<td>Delayed TKWakeups (Almost Ready)</td>
</tr>
<tr>
<td>frz</td>
<td>STATE_FROZEN</td>
<td>9</td>
<td>Frozen Thread via DosSuspendThread, DosCreateThread, DosExecPgm or DosSystemService</td>
</tr>
<tr>
<td>gsk</td>
<td>STATE_GETSTACK</td>
<td>10</td>
<td>TSD daemon is waiting for the page manager to page in a TSD</td>
</tr>
</tbody>
</table>
Notes:

STATE_READYBOOST is a modified ready state and never becomes the current state, instead a priority boost is applied and the state becomes STATE_READY.

See State field description of the .P Command for related information on thread states.

Pri
Thread priority in TCBPriority (TCB +0x168)
This is the current priority calculated by the scheduler based upon priority class (TCBPriClass (TCB +0x164)) priority class level (TCBPriLevel (TCB +0x165)) and priority boosts (TCBPriClassMod (TCB +0x166)). See Pri field description of the .P command for further information.

pTCB
Linear address of the TCB control block that represents the thread.

PriNextQ
The TCB address of the thread at the head of the next priority queue.
PriNextQ is taken from the TCBpTCBPriNextQ (TCB +0x170) double-word field.
If there are no other linked priority queues then TCBpTCBPriNextQ and TCBpTCBPriPrevQ point to this thread and PriNextQ and PriPrevQ are shown blank.
All TCBs not heading a priority queue have TCBpTCBPriNextQ and TCBpTCBPriPrevQ pointing to themselves.

PriNext and PriPrev is only of relevance to blocked and delayed threads.

PriPrevQ
The TCB address of the thread at the head of the previous priority queue.
PriPrevQ is taken from the TCBpTCBPriPrevQ (TCB +0x174) double-word field.
If there are no other linked priority queues then TCBpTCBPriNextQ and TCBpTCBPriPrevQ point to this thread and PriNextQ and PriPrevQ are shown blank.
All TCBs not heading a priority queue have TCBpTCBPriNextQ and TCBpTCBPriPrevQ pointing to themselves.

PriNext and PriPrev is only of relevance to blocked and delayed threads.

PriHigh
The TCB address of the next higher priority thread within this priority queue.
PriHigh is taken from the TCBpTCBPriHigher (TCB +0x178) double-word field.
If there are no higher priority threads on this priority queue then TCBpTCBPriHigher points to this thread and PriHigh is shown blank.

PriLow
The TCB address of the next lower priority thread within this priority queue.
PriLow is taken from the TCBpTCBPriLower (TCB +0x17c) double-word field.
If there are no lower priority threads on this priority queue then TCBpTCBPriLower points to this thread and PriLow is shown blank.

PriNext
The TCB address of the next thread of the same priority within this priority queue.
PriNext is taken from the TCBpTCBPriNext (TCB +0x180) double-word field.
If there are no other threads of the same priority on this priority queue then TCBpTCBPriNext and TCBpTCBPriPrev point to this thread and PriNext and PriPrev are shown blank.

PriPrev
The TCB address of the previous thread of the same priority within this priority queue.
**Scheduler Priority Queues**

Threads are linked in structures called **Priority Queues** or **PriQs**.

Priority queues are a double-linked list of thread priority groups. Each group is a double-linked list of threads of the same priority.

Six chain pointers are used for the links of a **PriQ**:

- **TCBpTCBPriHigher** (TCB + 0x178)
- **TCBpTCBPriLower** (TCB + 0x17c)
- **TCBpTCBPriNext** (TCB + 0x180)
- **TCBpTCBPriPrev** (TCB + 0x184)

By default, these chain pointers are set to point to their own TCB.

**TCBP** and **TCBP** link the heads of each priority group.

**TCBP** and **TCBP** link the TCBs within each priority group.

A number of **PriQs** are defined. Each is anchored from a global symbol pointer:

- **_ptcbPriQTSD**
  - Anchor for all threads in **tsd** state.
- **_ptcbPriQRunner**
  - Anchor for all threads in **run** state. At most this contains one TCB.
- **_ptcbPriQReady**
  - Anchor for all threads in **rdy** state.
- **_ptcbPriQGetStack**
  - Anchor for all threads in **gsk** state.
- **_ptcbPriQBadStack**
  - Anchor for all threads in **bad** state.
- **ptda_pTCBPriQCritSec**
  - Anchor per-process for all threads within a process in **crt** state.

**Notes:**

For the current process **ptda_pTCBPriQCritSec** (PTDA +0x2e4) is also a global symbol. Out of current context it can be located relative to the process' PTDA address.

The TCB address of the thread that has entered critical section is saved in **ptda_pTCBPriQCritSec** (PTDA +0x2e0)

Sleeping threads are queued on priority queues but in a manner to favour wake-up processing. The **Block-Id** is hashed to form an index into a table of **PriQ** anchors. The table is located at global symbol:

- **_aptcbSleep**

Each anchor points to a chain of **PriQs** of threads sleeping on the same Block-Id. The head TCB of each **PriQ** within a hashed chain is doubly linked from:

**TCBP** (TCB + 0x170)
Threads that happen to sleep on the same Block-Id as a multi-wake-up block-id are guaranteed not to be put in the same chain as the multi-wake-up threads.

When multi-wake-up threads wake their entire sleeping PriQ is moved to a chain of PriQs for threads in dly state. The delayed thread PriQ is anchored from global symbol:

```c
_ptcbPriQDelayed
```

Since `ptcbPriQDelayed` anchors a chain of PriQs, the head of each PriQ is doubly-linked using `TCBpTCBPriQNextQ` and `TCBpTCBPriQPrevQ`.

### .PU - Display Thread User Space Information

Display thread user space summary information for all (active) threads.

**Syntax:**

```
.PU
```

```
  #
  * slot
```

**Parameters:**

`slot`

Display user information for thread slot `slot`.

The following short-hand may be used for the slot number:

* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:
  ```c
  _TaskNumber
  ```

* The debugger default thread slot. This defaults to the current slot unless overridden by the `.S` command.

If no slot number is given then all thread slots are displayed in slot number order.

**Results & Notes:**

The `.PU` command locates each thread's TCB from the thread slot table, the linear address of which is given by global variable:

```c
_papTCBSlots
```

or by traversing the process tree using `TCBpTCBNext` (TCB +0x14), `TCBpPTDA` (TCB +0x08) and `ptda_pTCBHead` (PTDA + 0x20) fields.

Output from the `.PU` command appears in tabular form as follows:

```
Slot  Pid  Ord  pPTDA    Name     pstkframe  CS:EIP        SS:ESP     cbargs
0001  0001 0001 ffe3c7d4 *ager    ffe3bf54 1e30:00001794 0030:0000a402 0000
0002  0001 0002 ffe3c7d4 *tsd
0003  0001 0003 ffe3c7d4 *ctxh
0004  0001 0004 ffe3c7d4 *kdb
0005  0001 0005 ffe3c7d4 *lazyw
0006  0001 0006 ffe3c7d4 *asyncr
0008  0002 0001 7b9e4020 pmshel1 7b7df4c d02f:0000272d 001f:0003f8b8 0008
  *000a  0002 0002 7b9e4020 pmshell 7b7db44 d087:00003413 bffe:00007a6 0010
  000b#  0002 0003 7b9e4020 pmshell 7b7ddf48 d087:0000351a bffe:00000fc0 000c
```
Each of the fields has the following meaning:

**slot**

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

* to the left to signify the current (or last) dispatched thread.
# to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

**Slot** may be found in the TCBNumber (TCB + 0x2) field of the TCB

**Pid**

The process id this thread slot is assigned to.

**Ord**

The thread ordinal for this thread slot. This is the unique thread id assigned to the thread within the process to which it belongs.

**Ord** is taken from the TCBOrdinal (TCB+ 0x0) field of the

**pPTDA**

Linear address of the control block representing the process to which this thread belongs. The address is taken from TCBpPTDA (TCB+0x08).

**Name**

The name of the primary executable running in this process.

See name field description of the .P command for further information.

**pstkframe**

The address of the ring 0 stack frame that saved the user (ring 2 or ring 3) registers at the last transition to ring 0. For internal threads that have never run in ring 2 or ring 3 or for the currently running ring 3 thread this field will appear blank.

The address for the user stack frame is taken from TCB_pFrameBase (TCB + 0x3c). See .R command for further information on displaying registers saved in the user stack frame.

**CS:EIP**

The user (ring 2 or ring 3) CS:EIP saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the TSD for this thread is paged out. See .I command for information on paging in a TSD.

**SS:ESP**

The user (ring 2 or ring 3) SS:ESP saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the TSD for this thread is paged out. See .I command for information on paging in a TSD.

**cbargs**

The user (ring 2 or ring 3) cbargs saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the TSD for this thread is paged out. See .I command for information on paging in a TSD.

--------------------------------------------

**.R - Display User's Registers**

Display the user registers for a given thread slot. Set default addresses for E command, D command, K command and U command.

Applicable to the Dump Formatter only, the default addressing mode is not set according to the VM flags of the EFLAGS register but is assumed always to be in protect mode. This has been corrected from fix pack 29 of Wapr 3.0 and base Warp 4.0.

**Syntax:**
Parameters:

slot

Display user registers for thread slot \textit{slot}. This option is valid \textbf{only} under the Kernel Debugger.

The following short-hand may be used for the slot number:

* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

\_TaskNumber

# The debugger default thread slot. This defaults to the current slot unless overridden by the \texttt{.S} command.

If no slot number is given then the debugger's default slot number is assumed.

Results & Notes:

Registers are displayed and register mnemonics are assigned the values displayed for use in address expressions and operands of other Kernel Debugger and Dump Formatter commands.

The register information is obtained as follows:

Under the Kernel Debugger, if the displayed slot is the current system slot and the system is not in kernel mode (that is, \texttt{Indos} \texttt{\neg=} \texttt{1}) then the hardware register values save by the debugger are displayed.

Otherwise the registers are extracted from the from the ring 0 stack frame base pointed to by \texttt{TCB\_pFrameBase} (TCB + 0x3c) for the thread slot in question.

The ring 0 stack frame base is created when the threads makes a transition from ring 2 or 3 to ring 0. This happens for a variety of reasons, such as issuing a call gate, trapping, pre-emption, interrupt, etc.. The format of the stack frame base depends on the reason for the ring 0 transition. \texttt{TCB\_pcriFrameType} (TCB + 0x38) points to the CRI, which contains a table of RIPs. Each RIP entry is associated with a specific hardware register. The RIP contains the offset and length of the associated register saved in the stack frame base. See \textit{Client Register Information and Stack Frames} for details of the \texttt{CRI} and \texttt{RIP} formats.

Note:

If the thread has never run out of kernel mode, as is the case with some internal threads, then the \texttt{CRI} is never updated. The \texttt{.R} command is not able to format the user registers. For these threads the \texttt{R command} should be used, but only when the thread in question is the current system thread. Because the \texttt{R command} is an alias to the \texttt{.R} under the Dump Formatter, there is no way to display the current registers for an internal thread under the Dump Formatter. The only recourse is to display the \texttt{TSD} for the thread and attempt to unravel the stack manually.

If an invalid thread slot number is given the the Kernel Debugger issues the following message: prompted with the command prompt.

\texttt{Invalid task number: mnnn}

The format of the \texttt{.R} command output depends on whether the \texttt{RT command} has been used to toggle register display to full or short form and also whether the \texttt{Y 386ENV command} has been used to toggle register interpretation into 286 or 386 mode. Examples of the various forms follow:

```c
##rt
##.r 2c
eax=f110099f ebx=00000001 ecx=0133fe4c edx=00000007 esi=0133ffe0 edi=00000001
eip=00000626 esp=0133fe20 ebp=0133fe88 iopl=2 -- -- nv up ei ng nz na pe nc
```

```c
cs=002f ss=099f ds=0053 es=0053 fs=150b gs=0000 cr2=1581928c cr3=001d0000
```
Following the formatted register display, one line of disassembled code is displayed at the default disassembly address. See the U command for details on disassembling code.

Each of the fields has the following meaning:

**General Registers**
These comprise the following registers:

- ax, bx, cx, dx, sp, bp, si, di
- eax, ebx, ecx, edx, esp, ebp, esi, edi

Each is displayed with its value in hexadecimal.

**Segment Registers**
These comprise the following registers:

- cs, ds, es, fs, gs, ss

Each is displayed with its selector value in hexadecimal.

**Instruction Pointers**
These comprise the following registers:

- ip & eip

Each is displayed with its value in hexadecimal.

**Flag registers**
These comprise the following registers:

- flags & eflags

These have their bit setting interpreted as follows:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Flag Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>VM Virtual 8086 Mode (EFLAGS only)</td>
</tr>
</tbody>
</table>
16 0  RF  Resume Flag – Disable Debug
   Exceptions (EFLAGS only)
14 1  NT  Nested Task
11 1  OV  Overflow
11 0  NV  ¬Overflow
10 1  DN  Direction Down
10 0  UP  Direction Up
  9 1  EI  Enable Interrupts
  9 0  DI  Disable Interrupts
  7 1  NG  Negative Sign
  7 0  PL  Plus Sign
  6 1  ZR  Zero Result
  6 0  NZ  Non-zero Result
  4 1  AC  Auxiliary Carry
  4 0  NA  ¬Auxiliary Carry
  2 1  PE  Parity Even
  2 0  PO  Parity Odd
  0 1  CY  Carry
  0 0  NC  ¬Carry

Bits 12 and 13 are the I/O Privilege Level bits. These are formatted as \texttt{iopl=level}.

Flags 14, 16 and 17 when reset are indicated by --.

\textit{Memory Management Registers}

\begin{itemize}
\item\texttt{gdtr=xxxxxxxx yyyy}  \quad \text{Global Descriptor Table Register base address (xxxxxxxx) and limit (yyyy)}
\item\texttt{idtr=xxxxxxxx yyyy}  \quad \text{Interrupt Descriptor Table Register base address (xxxxxxxx) and limit (yyyy)}
\item\texttt{ldtr=xxxx}  \quad \text{Local Descriptor Table Register GDT selector (xxxx).}
\item\texttt{tr=xxxx}  \quad \text{Task Register GDT selector (xxxx).}
\end{itemize}

\textit{Control Registers}

\begin{itemize}
\item\texttt{cr0=}  \quad \text{System control flags and Machine Status Word.}
\end{itemize}

These have their bit setting interpreted as follows:

\begin{itemize}
\item\texttt{Bit} \quad \text{Value} \quad \text{Flag Description}
   \item 31 1  PG  Paging Enabled
   \item 4 1  ET  Extension Type Flag – x87 support
   \item 3 1  TS  Task Switch Flag
\end{itemize}
Reset flag bit are shown with --.

\textbf{cr2=}

Page fault linear address.

\textbf{cr3=}

Page Directory Base Register (PDBR).

\textbf{Debug Registers}

\textit{dr0} to \textit{dr3}

These are formatted as follows:

\begin{verbatim}
  dr0=11111111 gixnb
  dr1=11111111 gixnb
  dr2=11111111 gixnb
  dr3=11111111 gixnb
\end{verbatim}

where \textit{liiiii} is the breakpoint linear address and \textit{gixnb} are \textit{dr7} and \textit{dr6} related flags.

The flags have the following interpretations:

- \textit{g} \quad G indicates a globally enabled breakpoint.
- \textit{l} \quad L indicates a locally enabled breakpoint.
- \textit{x} \quad E indicates an execute breakpoint.
- \textit{r} \quad R indicates a read breakpoint.
- \textit{w} \quad W indicates a write breakpoint.
- \textit{n} \quad The number of bytes tested (1, 2 or 4).
- \textit{b} \quad B indicates a that a debug exception was generated that matched this breakpoint. This is the \textit{Bn} value of \textit{dr6}.

- \textit{b} \quad B indicates a that a debug exception was generated that matched this breakpoint. This is the \textit{Bn} value of \textit{dr6}.

\textbf{dr6=}

The control bits 13-15 are interpreted as follows:

\begin{tabular}{|c|c|c|c|}
  \hline
  Bit & Value & Flag Description \\
  \hline
  15 & 1 & BT & Breakpoint triggered on task switch \\
  14 & 1 & BS & Breakpoint triggered on single step. \\
  13 & 1 & BD & Breakpoint on debug register access/update. \\
  \hline
\end{tabular}

Flag bits not set are indicated by --

\textbf{dr7=}

The control bits 8 and 9 are interpreted as follows:
<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Flag Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>GE  Exact data matching enabled for global breakpoints</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>LE  Exact data matching matching for local breakpoints</td>
</tr>
</tbody>
</table>

Flag bits not set are indicated by --

**Test Registers**

\[\text{tr6=lllll \ v=d=dd \ u=uu \ w=ww \ c=c}\]

lllll is the linear page address.

\(\text{v}\) is \text{tr6} flag bit 11, the valid bit.

\(\text{dd}\) are \text{tr6} flag bits 10 and 9.

\(\text{uu}\) are \text{tr6} flag bits 8 and 7.

\(\text{w}\) are \text{tr6} flag bits 6 and 5.

\(\text{c}\) is set as follows:

\(\text{r}\) \quad \text{tr6} flag bit 0 set. TLB read command.

\(\text{w}\) \quad \text{tr6} flag bit 1 reset. TLB write command.

\[\text{tr7=ppppp \ ht=h \ rep=r}\]

ppppp is the \text{tr7} physical frame address.

\(\text{h}\) is flag bit 4 value. This is the hit or PL bit.

\(\text{r}\) are \text{tr7} flag bits 3 and 2. These are the report or REP bits.

The following INTEL(R) publications should be consulted for definitive information on processor registers:


Pentium(TM) Processor User’s Manual

---------------------------------------------

.REBOOT - Restart the System

Restart the system.

**Syntax:**

```
.REBOOT
```

**Parameters:**

None
Results & Notes:

DosHlp service DosHlpReboot is called to restart the system. No system shutdown processing, whatsoever, is performed.

This command is not available to the Dump Formatter!

--------------------------------------------

.S - Set or Display Default Thread Slot

Set or display the Kernel Debugger's and Dump Formatter's default thread slot.

This command affects the default operation of the following:

D command
E command
U command.
.I command
.K command
.P command
.PB command
.PQ command
.PU command
.R command

Syntax:

.S
.S slot

Parameters:

slot

Set the default thread slot to slot.

The following short-hand may be used for the slot number:

* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

_TaskNumber

If no slot number is given .S displays the current thread slot number in message:

Current task number: nnnn

where nnnn is the thread slot number.

S

Set current ESP, EBP, SS, CS and EIP registers to those of the Dispatcher.

This option sets these registers as if the thread context had just been switched by the OS/2 Dispatcher. The R command will show the thread in kernel mode, about to be run.

No actual updating of register values takes place. Only default values are effected.

The new defaults are derived as follows:

ESP taken from TSDKernelESP (TSD + Disp value of .P command output.)
EBP               taken from TSDUserSSPad (TSDKernelESP - 2)
SS                selector 30 (TASKAREA segment).
CS                Selector 170 (DOSHIGH32CODE segment).
EIP               label pgSwitchRet.

This option is not available to the Dump Formatter.

Note:

The intent of this option is to simulate the correct value of the ring 0 stack selector for the default thread. This
is only safe to use in commands that make explicit reference to the stack selector, for example:

.sS 21
A
.DD SS:ESP

If an indirect reference is made to selector 30, for example by referring to a symbol from the TASKAREA
segment then the adjustment to the default slot is not made. For example:

.DW JFN_Table 114

will only display the JFT for the current thread slot. To display the JFT for another slot requires the following
technique:

.DW %ptda_address+JFN_Table-ptda_start 114

Results & Notes:
The .S command sets certain default values such that the view of the user’s space in the new default slot is as if the the thread context had
switched. Linear and LDT selector based addresses will be accesses correctly by the Dump Formatter and Kernel Debugger. However,
certain system data that are updated by a context switch are not changed and continue to display in the system's current thread context.
These items include:

Task Register (TR)
GDT descriptor table entries for selectors 28, 30, 38 and 150b
Current TSS ring 0, and ring 2 stack selectors and pointers.
Global and System copy of the Current Local Information Segments.
The Thread Local Memory Area and Local Information Segment mapped by LDT descriptor dfff.

Note:

Descriptor dfff maps a global shared memory object, but it's data is copied from the incoming PTDA and TCB when a
context switch occurs. This achieves the effect of thread local memory.

--------------------------------------------

.SYSDUMP - Force a System Dump and Restart the System

Dump and Restart the system.

Syntax:

.SYSDUMP
Parameters:

None

Results & Notes:

The Kernel Debugger gives control to RASRST to force a system dump. The dump drive will be that specified in variable: DumpDevice, which defaults to A. This may be modified by using the E command.

The current CPU register values as displayed by the R command may be found from symbol _RegSA and formatted using the PMDF REXX EXEC DR. See Forcing a System Dump from the Kernel Debugger for further information. No system shutdown processing, whatsoever, is performed.

This command is not available to the Dump Formatter!

--------------------------------------------

.T - Dump the System Trace Buffer

Dumps the system trace buffer.

Syntax:

.T count MAJ=mm MIN=nn S filespec

Parameters:

count

The number of trace entries to print, starting with the most recent. If not specified then the entire trace buffer will be dumped.

MAJ=mm

Specifies that only trace events with major code mm should be displayed.

See System Trace Facility - Major Code Assignments for a information on the deployment of trace major and minor codes in OS/2.

Warning: The Kernel Debugger may fail to process the MAJ= parameter correctly. Under some circumstances the debug kernel may hang. Use this option advisedly!

MIN=nn

Specifies that only trace events with minor code nn should be displayed.

This option required the specification of a major code using the MAJ= parameter.

See System Trace Facility - Major Code Assignments for a information on the deployment of trace major and minor codes in OS/2.

Warning: The Kernel Debugger may fail to process the MAJ= parameter correctly. Under some circumstances the debug kernel may hang. Use this option advisedly!

S

Specifies that the trace buffer should be saved to a file named in filespec.

This option is only available to the Dump Formatter.
The saved trace file may be subsequently formatted using the OS/2 TRACFMT command.

filespec

The file specification for the saved trace buffer.

The filespec may be fully qualified. The path defaults to the current directory.

Results & Notes:

The trace is activated using the OS/2 TRACE command.

If the trace is not active then the following message is generated:

    Trace not on

The trace buffer is allocated in a single segment (STDA) whose selector may be located from global symbol ras_stda_addr. The STDA is a circular buffer whose entries are recorded in reverse order. The header gives the offsets to the first, last and current entries. The format of the trace buffer is described under System Trace Data Area.

The major codes being traced are recorded in a bit string located at label ras_mec_table. Each active major code has its corresponding bit set.

The minor codes being traced are recorded in a bit string whose selector is located at label ras_min_table. The minor code table contains 32 byte entries, each corresponding to a major code. Each bit of each entry corresponds to a minor code within the major. If the bit is set then the minor code is traced.

When tracing by Pid is active then the ptda_rasflag (PTDA +0x39a) is set to 0xff.

The status of system tracing is recorded in status byte at label ras_systr_flags. The following flags are defined:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_TRCAVAIL</td>
<td>0x80</td>
<td>System Trace Available</td>
</tr>
<tr>
<td>RF_TRCPAUSED</td>
<td>0x40</td>
<td>Trace paused</td>
</tr>
<tr>
<td>RF/TRCPID</td>
<td>0x20</td>
<td>Trace by PID</td>
</tr>
<tr>
<td>RF_TRCERRCOUNT</td>
<td>0x10</td>
<td>Tracing until error count</td>
</tr>
<tr>
<td>RF_TRCSUSPEND</td>
<td>0x08</td>
<td>Suspend due to error count</td>
</tr>
<tr>
<td>RF_TRCMINORCD</td>
<td>0x04</td>
<td>Tracing by Minor Code</td>
</tr>
</tbody>
</table>

Under the kernel debugger the system trace buffer cannot be saved directly. However by setting the RF_TRCPAUSED bit in ras_systr_flags flag byte, the trace may be suspended and saved at a later time by using one of the system trace utilities (TRSPOOL, TRACEFMT or TRACEGET) or from a system dump. When setting RF_TRCPAUSED be certain to OR in the flag bit.

.T command output appears as follows (see note at the end of this section for information on recent changes to the format of trace output):

```
MAJ=04  MIN=0089  PID=0006  CONTEXT=KERNEL:PROTECT
MAJ=06  MIN=008c  PID=0000  CONTEXT=KERNEL:PROTECT
  00 00
MAJ=06  MIN=000c  PID=0000  CONTEXT=KERNEL:PROTECT  TS=1336
      08 00
MAJ=04  MIN=0009  PID=0006  CONTEXT=KERNEL:PROTECT
MAJ=04  MIN=0089  PID=0006  CONTEXT=KERNEL:PROTECT
```

Each of the fields is defined as follows:

MAJ=

The traced event major code.

MIN=

The traced event minor code.

PID=

The current Pid when the event occurred. See .P command for information on displaying active Pids.
The system and processor context under which the event was traced.

**system context may be:**

**KERNEL**
If the trace record was created internally by a kernel routine.

**API**
If the trace record was created externally by use of the DosDynamicTrace or DosSysTrace APIs.

See Dynamic Trace Customiser for information on creating dynamic trace records (via DosDynamicTrace).

See DosSysTrace (Static Trace Event Recording) for information on creating static trace records.

**processor context may be:**

**PROTECT**
If the trace record was created when the system was running in protect mode.

**REAL**
If the trace record was created when the system was running in real mode.

**TS=hhss**
The system timestamp where $hh$ is 100th seconds and $ss$ is seconds.

The timestamp is taken from the Global Information Segment (GISEG+0xa). It is only recorded in the trace record if the time has changed since the previous timed stamped record was recorded.

*Note:* TRACEFMT treats this value as a word length fixed number of two decimal places.

**trace data**
Additional trace data.

A trace event may be accompanied with additional trace data, in which case it is dumped in hexadecimal and ASCII format on the following line.

Related information on the system trace facility may be found in:

- System Trace Facility
- Dynamic Trace Customiser
- OS/2 Command reference - TRACE Command
- OS/2 Command reference - TRACEFMT Command
- OS/2 Command reference - TRACEBUF CONFIG.SYS statement

**New Trace Format**

From OS/2 2.11 fix pack 91 and OS/2 3.0 fix pack 8, the system trace has been enhanced to include more useful timestamp information.

The Kernel Debugger and Dump Formatter were updated in fix packs 16 (OS/2 3.0) and 105 (OS/2 2.11) to take account of the new format.

**Warning:**

The use of the .T command after the new trace format was implemented, but before the Kernel Debugger and Dump Formatter were updated, caused the Kernel Debugger and Dump Formatter to trap.

The following is an example of the new format:

```
# Trace On at 0000,0000,0000,0000,0000,0000,0000
Trace Off at 0000,0000,0000,0000,0000,0000,0000
MAJ=03 MIN=0009 PID=0000 CONTEXT=KERNEL:PROTECT TS=3611,382e
 00 00 00 00 bd 55 f5 ff 60 01 00 00 02 00 01 00 ....=Uu.'.......
MAJ=03 MIN=000f PID=0000 CONTEXT=KERNEL:PROTECT TS=3611,382e
 00 00 cc cc f1 27 00 00 10 00 00 06 02 01 00 ..LLq'...........
  c8 3c f2 ab ...H<
MAJ=03 MIN=0008 PID=0017 CONTEXT=KERNEL:PROTECT TS=3611,252d
 00 00 00 00 93 86 e5 1b 5b 00 00 00 02 22 01 00 ........e.[...."..
MAJ=03 MIN=0008 PID=0017 CONTEXT=KERNEL:PROTECT TS=3611,222d
```
The formatted trace is headed by a pair of time-stamps that give the time tracing was initiated and terminated. These are of the form:

YYYY,xxMM,xxDD,xxHH,xxmm,xxss,xxhh

where:
YYYY is years,
MM is Months
DD is Days,
HH is hours,
mm is minutes,
ss is seconds.

hh is 1/100ths seconds,
xx ignore.

The time-stamp of each trace record is now shown as a pair of word values of the form:

TS=MMHH,hhss

where
MM is minutes,
HH hours,
hh 1/100ths seconds and
ss seconds.

Note:
The byte reversal occurs because the time values are originally byte values which are displayed as words.

System Trace Facility - User Guide

The OS/2 Trace facility is an important RAS mechanism within the OS/2 product. It allows specific events within the operating system, in system extensions and in applications to be recorded in a circular System Trace buffer. Software developers can create tracepoints that are used to monitor the execution of their software modules. A tracepoint is in essence a "window" that can be used to "peek" into a software module whenever that module reaches the state that corresponds to the tracepoint.

The OS/2 Trace facility includes three important utility programs:

**TRACE**
The OS/2 Trace control utility is used to control (that is, enable and disable) the tracing of individual events.

**TRCUST**
The Dynamic Trace Customiser is used to define both dynamic tracepoints for .DLL modules and trace formatting files for use by TRACEFMT utility.

**TRACEFMT**
The Trace Formatter is used to format the contents of the system trace buffer for viewing. It may also be used for saving both formatted and unformatted trace data in a disk file and therefore may be used with the Dump Formatter
The TS command to format a system trace buffer that has been captured in a system dump.

The following topics are discussed in this chapter:

Dynamic versus Static Trace
Guidelines for Defining Tracepoints

--------------------------------------------

Dynamic Versus Static Trace

OS/2 supports two types of tracepoints:

• Static tracepoints
• Dynamic tracepoints

Both types of tracepoints can be used to monitor the execution of a software module. They differ in their style of execution.

Static tracepoints are, essentially, in-line function calls that are always present. The OS/2 user has the ability (through the use of the OS/2 TRACE utility) to indicate that a particular static tracepoint is "enabled". Until a static tracepoint is enabled, the actual tracepoint logic is "branched over" and not executed. Once "enabled" the static tracepoint logic is executed.

Dynamic tracepoints do not normally reside within the software modules to which they correspond. They are "patched in" when the OS/2 user uses the TRACE utility to "enable" a dynamic tracepoint.

This implies that a dynamic tracepoint does not burden a software module with any execution overhead until the tracepoint is "patched in" by the TRACE utility. There is a small continual performance overhead associated with a static tracepoint because the tracepoint must always check to see whether it is currently enabled. On the other hand, the dynamic tracepoint mechanism is costlier in operation than the static tracepoint operation because its "patching" mechanism is built upon the OS/2 breakpoint mechanism. Also, there are some software modules (for example, device drivers) that cannot use dynamic tracepoints. Most of the operating system tracepoints listed in the Trace Reference are implemented as dynamic tracepoints.

All Trace event records include a major trace code and a minor trace code which identify the event which is being recorded. When the user uses the TRACE utility to control a specific tracepoint, static tracepoints are identified by a combination of major and minor trace codes. Dynamic tracepoints are identified by a combination of software module name and minor trace code.

The following general categories of events are traced within OS/2:

1. External application program interfaces (APIs)
2. Internal interfaces
3. Other internal events

--------------------------------------------

Guidelines for Defining Tracepoints

The following subsections provide guidelines for programmers who are adding tracepoints to their software modules.

• Assignment of Major and Minor Codes
• Trace Event Call Location
• Trace Event Parameters

--------------------------------------------

Assignment of Major and Minor Codes
Component and subsystem developers who assign their own major and minor trace codes should follow the following conventions:

1. Minor codes should be assigned in the range 0001H-FFFFH. Minor code 0000H is reserved and should not be used.

2. An interface/event which requires both a pre-invocation trace and post-invocation trace of the event should share the same major code. The minor code assigned for the pre-invocation trace should be in the range 0001H-7FFFH. The minor code for the post-invocation trace of the event should be in the range 8001H-FFFFH and should be assigned by turning on the high order bit of the pre-invocation minor code. For example:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Major Code</th>
<th>Minor Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>KbdCharIn Pre-Invocation</td>
<td>0011H</td>
<td>0001H</td>
</tr>
<tr>
<td>KbdCharIn Post-Invocation</td>
<td>0011H</td>
<td>8001H</td>
</tr>
</tbody>
</table>

3. The Trace Definition Files for dynamic tracepoints should use the following TYPE definitions when defining tracepoints. Other TYPEs or GROUPs may be defined as desired but these should be used as a minimum.

   - TYPE=(PRE,API) - pre-invocation of external interface
   - TYPE=(POST,API) - post-invocation of external interface
   - TYPE=(PRE,INT) - pre-invocation of internal interface
   - TYPE=(POST,INT) - post-invocation of internal interface

---

Trace Event Call Location

Trace events, each with a unique minor code, should be placed at the following locations within a component or subsystem performing a service as a result of an external API call:

1. Pre-Invocation: At the beginning of any external interface before performing any actions or changing any data. This is considered pre-invocation of a service.

2. Post-Invocation: Before returning from performing any service through an external interface. This is considered post-invocation of a service, and is significant since:
   a. Information may be returned to the caller which is of interest for problem determination
   b. It causes a "service completion" event to be added to the trace buffer which may be paired with the pre-invocation trace event

3. Other: Events within a component should be considered as points of interest for tracing when they may be used for problem determination within the component or subsystem. These include:
   a. Significant changes in the state of the machine.
   b. Significant changes in the state of the system. System state changes are usually those which affect system data structures. These structures are those which are referenced by more than one component or which represent significant system resources.
   c. Allocation or deallocation of a resource managed by a component performing a service

This detail is determined and assigned by each component or subsystem owner.

---

Trace Event Parameters

In general, interface parameters should be passed with the trace call. Interfaces to performance sensitive code may require that only a few of the most important parameters, if any, be passed with the trace call.
1. Pre-Invocation:
   a. Call by value parameters set by the caller should be passed with the trace call
   b. Call by reference parameters to simple data (BYTE, WORD, DWORD) set by the caller should be passed and copied with the trace call
   c. Call by reference parameters to character string data set by the caller should be passed and copied with the trace call
   d. Call by reference parameters to complex data (structures other than character strings) should be handled by passing a subset of the most important data in the structure if the entire structure can not/need not be recorded

2. Post-Invocation:
   a. A return code should always be returned from a post-invocation trace call. If the return code is not zero, then any additional parameters that are traced may not be valid.
   b. Call by value parameters returned by the service should be passed with the trace call
   c. Call by reference parameters to simple data (BYTE, WORD, DWORD) returned by the service should be passed and copied with the trace call
   d. Call by reference parameters to character string data returned by the service should be passed and copied with the trace call
   e. Call by reference parameters to complex data (structures other than character strings) should be handled by passing a subset of the most important data in the structure if the entire structure can not/need not be recorded

3. Other:
   a. Significant data relating to the event
   b. Address of a control block; Id or address of a resource

--------------------------------------------
Dynamic Trace Customiser (TRCUST) - Reference

OS/2 provides a mechanism by which developers may dynamically apply tracepoints in their module at run time. This method eliminates all overhead of tracing when tracing is disabled. It also allows the developer to add tracepoints without modifying source code. This reduces the possibility that adding a tracepoint will induce errors into one's code. OS/2 needs a binary file, for each module being dynamically traced, which defines the tracepoints for the module.

Note:

Information given here refers to the following versions of the system tracing tools except where explicitly noted:

TRCUST 3.06 or higher
TRACE 2.4 or higher
TRACEFMT 2.4 or higher
TRSPPOOL 4.2 or higher
TRACEGET with OS/2 Warp V3.0 FP35, OS/2 Warp V4.0 FP10 or OS/2 Warp E-Server or higher
DTRACE 4.3 or higher
TRSPPOOL 4.1 or higher
TFFLST 1.1 or higher
TDFLST 1.7 or higher
DEBDEL 1.0 or higher
Note:

There are certain restrictions on the use of dynamic trace which should be noted. These are:

1. Dynamic tracepoints may be applies to any non-VDM protect-mode module. However, in order to apply tracepoint to non-DLL modules a version of the trace command shipped with OS/2 Warp V3.0 fix pack 32, OS/2 Warp V4.0 fix pack 1 or OS/2 Warp E-Server is required.

2. Dynamic tracepoints may be applied to routines that run at interrupt time only after fix pack 35 for OS/2 Warp V3.0, fix pack 1 of OS/2 Warp V4.0 or OS/2 Warp E-Server.

3. A dynamic tracepoint cannot be applied to a module running under the Kernel Debugger that has a Breakpoint in place at the same location as the trace point.

The Trace Customizer (TRCUST) converts tracepoint definitions from a trace source file (TSF) into dynamic tracepoints for the trace definition file (TDF), and into formatting rules in the trace format file (TFF).

TRCUST provides a high-level access to Dynamic Trace, particularly suitable for use by developers. The associated DTRACE tool provides a very low-level interface to Dynamic Trace, which is suitable for attacking complex problems. Full details of the Dynamic Trace facility may be found in The Dynamic Trace Facility And The DTRACE Tool.

Definitions

.TSF  An ASCII file created by a developer which defines all dynamic tracepoints for a given module. TRCUST allows just one major code to be associated with a TSF.

.TDF  A binary file, created by TRCUST, using the .TSF file as input. This file defines all tracepoints in the module in a manner acceptable to OS/2. This is used by the Trace Utility, TRACE.

.TFF  A binary file also created by TRCUST using the .TSF file. This file defines how all tracepoints will be formatted. This is used by the Trace Formatter, TRACEFMT.

major code  A byte value used to identify the module being traced. TRCUST allows at most only one major code per TSF.

minor code  A word value used to uniquely identify each tracepoint.

GROUP  A value used to identify this tracepoint with tracepoints of the same category. Examples are MEM for memory management and PM for Presentation Manager. For an example of uses of groups, see the online help for the TRACE command.

TYPE  A value used to associate a subset of dynamic trace events within a module. Examples are PRE for pre-invocation, POST for post-invocation and API for API calls within a module. For an explanation and examples of uses of types, see the online help for the trace command.

Overview

File Naming and Location

Invoking the Trace Customizer

File Naming and Location

The TDF file name is the same as the module to be traced, but has a file extension of “TDF”. The TFF has a name of the form
"TRC00xx.TFF" where xx is the major code, for example, a module with major code 0xC2 will generate a TFF with the name "TRC00C2.TFF". This naming convention is used to allow TRACEFMT to dynamically generate the TFF name given only the major code.

TRCUST can be invoked to process a TSF or to combine several TFF files into a single TFF. For processing a TSF, TRCUST is given the name of a TSF, and optionally:

- the desired name of the resulting TDF
- the MAP file name
- the error message level

TRCUST will store the TSF tracepoint formatting specifications in the TFF file and if the tracepoint specified was not for a static tracepoint, the TSF tracepoint definition will be converted into the format required by the Trace Utility and stored in the TDF file. On errors, TRCUST will display appropriate messages, skip any tracepoint with errors in its definition, and continue processing the next tracepoint definition.

For combining TFF files that use the same major code, TRCUST is given the name of the file containing the TFF filenames to combine and the name of the file to contain the combined trace format statements.

TRCUST will issue an error message and abort processing under the following conditions:

- the TSF cannot be opened
- when combining TFF files, if any TFF input files cannot be read or if all TFF input files do not use the same major code
- when defining dynamic tracepoints, if the executable module to contain the tracepoints cannot be read
- the TDF, or TFF files cannot be written to
- an error in the header definition in the TSF
- a missing ending quote in the TSF

Note: TRCUST always returns 0 so that, when invoking it from a makefile, processing of the rest of the makefile can continue if TRCUST aborts.

Combine TFF files when several modules that use dynamic tracing use the same major code. The Trace Formatter can only use one TFF file per major code to get formatting information from. After the TSF file for each module is run through TRCUST to produce a TDF and TFF file, TRCUST can be invoked again, this time using the combine TFF files option and a file that only contains the paths to all the TFF files using the same major code. TRCUST will read all the TFF files. If all TFF files don't use the same major code, TRCUST will issue an error message and abort. TRCUST will read each trace format record from the TFF files and write them (in ascending order according to minor code) to the destination TFF file given.

Invoking the Trace Customizer

The Trace Customizer is a protect mode only program and must therefore be run under OS/2.

TRCUST operates in two possible modes:

- It may be invoked to combine TFF files,
- or to process a TSF.

Invoking TRCUST to Combine TFF files.

The syntax for combining TFF files is as follows:

[d:\[path]TRCUST [d:\[path]tffsource /C=[d:\[path]tffdest [/Wn]
where:

**TRCUST**

is the name of the Trace Customizer program. A drive and path may optionally be specified to explicitly define the location of the Trace Customizer program, otherwise the program is searched for in the current directory, followed by looking along the path defined by the PATH environment variable.

[d:][path]tffsource

is the name of a file containing fully qualified (including extensions) pathnames of TFF files to combine. Each TFF file must use the same major code and each filename in the tffsource file is separated by white space. This will combine all TFF files for the same major code into a single TFF file. If duplicate minor code format definitions are found, the first format definition for the minor code remains valid, the duplicates are discarded and a warning message is issued. If no path is provided the tffsource file is searched for in the current directory, followed by using the current value of DPATH.

[d:][path]tffdest

is the name of the trace format destination file to store the combined trace format definitions.

/Wn (optional)

is the level of error messages to be displayed, where n can be 0, 1, or 2. The possible message levels are shown below along with the messages that each displays:

0  fatal and severe messages
1  fatal, severe, and error messages
2  all (fatal, severe, error, and warning) messages

A message level of 2 is the default.

An example of a tffsource file for using the combine TFF files option of TRCUST is:

```
\TFF\PROG1\TRC00C2.TFF \TFF\PROG2\TRC00C2.TFF
\TFF\PROG3\TRC00C2.TFF \TFF\PROG4\TRC00C2.TFF
```

To invoke TRCUST to combine TFF files using the above file as input (assume filename is \TFF\PROG\TRC00C2) and output the combined format statements into file \TFF\PROG\TR\TRC00C2.TFF is:

```
TRCUST \TFF\PROG\TRC00C2 /C=\TFF\PROG\TR\TRC00C2.TFF
```

------------------------------

**Invoking TRCUST to Process a TSF file.**

The syntax for processing a TSF file is as follows:

```
```

where:

**TRCUST**

is the name of the Trace Customizer program. A drive and path may optionally be specified to explicitly define the location of the Trace Customizer program, otherwise the program is searched for in the current directory, followed by looking along the path defined by the PATH environment variable.
is the name of the trace source file. If no file extension is provided then an extension of TSF is assumed. If no path is provided the trace source file is searched for in the current directory, followed by using the current value of DPATH.

is the name of the trace definition file to store the dynamic tracepoint definitions. If not specified, the TSF filename is used with an extension of TDF. If no file extension is provided then an extension of TDF is assumed.

allows duplicate minor codes to be used. This is useful where there is no need to distinguish different tracepoints which create records of the same format. For example, multiple return points from a subroutine.

allows case-insensitive references to MAP file symbols to be used in TSF TRACE statements.

specifies the load module path and file name to be read by TRCUST. If not specified TRCUST uses the specification in the MODNAME statement of the TSF. If no path information is given in the MODNAME statement TRCUST assumes uses current directory and DPATH to locate the load module.

MODNAME is required by the TRACE command to allow it to load and determine the traced module's handle. /L is useful in cases where the load module is built using a name that differs from the installed name or where path information is required on the MODNAME statement for the TRACE command which differs from the directory used when the TDF is built. See also the /P switch for TRCUST.

defines mapfile as the MAP file for this module. The name may be qualified by a drive/directory, otherwise it will be searched for in the current directory followed by the path specified by the DPATH environment variable. If specified as an option, the MAP file must exist and the filename extension must be MAP or TRCUST will abort processing. The mapfile will only be used if a symbol is not found in the symbolic debug information stored in the executable module.

forces TRCUST to ignore debugging information even if present. This is provided for cases where

a) the user wishes to use MAP symbols in preference to debugging information.

b) the level of debugging information in not supported by TRCUST and errors are produces if it is used.

Debugging information is not a public standard. Some compilers may appear to emulate supported styles of symbolic debugging information while not actually doing so.

TRCUST supports the following debugging information styles:

- IBM C and C6 CodeView NB00 and NB02 styles.
- IBM CSet/2 HLL version NB03 style.
- IBM CSet/2++ and IBM VisualAge V1 to V3 HLL version NB04 styles.
- IBM ALP assembler Debugging Information (HLL version NB04).

Any compiler or assembler that conforms to the HLL NB00 - NB04 debugging information styles.

forces TRCUST to ignore module line number records but honor any other debugging information present. This option is provided for cases where line number information does not conform to supported specifications and causes unpredictable results. Currently TRCUST is limited to supporting no more then 64K of line number information. For exceedingly large source files this limit could be exceeded.

forces TRCUST to ignore debugging information even if present. This is provided for cases where

a) the user wishes to use MAP symbols in preference to debugging information.

b) the level of debugging information in not supported by TRCUST and errors are produces if it is used.

Debugging information is not a public standard. Some compilers may appear to emulate supported styles of symbolic debugging information while not actually doing so.

TRCUST supports the following debugging information styles:

- IBM C and C6 CodeView NB00 and NB02 styles.
- IBM CSet/2 HLL version NB03 style.
- IBM CSet/2++ and IBM VisualAge V1 to V3 HLL version NB04 styles.
- IBM ALP assembler Debugging Information (HLL version NB04).

Any compiler or assembler that conforms to the HLL NB00 - NB04 debugging information styles.
allows the path information specified with MODNAME to be retained in the TDF. The default is to strip path information from the module name. With no path information the TRACE command will rely on PATH, LIBPATH or the command line specification to the TRACE command to locate the traced module.

allows a tracepoint to be placed after the MOV (E)BP,(E)SP instruction on entry to a routine where local variable references will be valid. This is intended for use where local data references are made from the TSF, but the tracepoint is defined by reference to the routine's MAP file symbol name. In general it is better to use number references since /PREINV relies on the style of code generated by the compiler used. See name specification of the TP statement for information on difference between MAP file symbol references and symbolic debugging information references.

Note: PREINV does not search for the equivalent ENTER instruction.

allows major code 0 trace definitions to be generated.

Note: Major code 0 is reserved by the system for system trace internal processing. The /RAS option should not be used except for the express purpose for which it is intended.

specifies the mode by which the RETEP keyword of the TSF TRACE statement will operate. Eight modes of RETEP operation are defined as follows:

0 Disallow RETEP tracepoints.
1 determine the RETEP tracepoint directly from CodeView symbol records only.
2 use mode 1 then search for a LEAVE+RET or POP EBP instruction sequences near the end of the traced routine.
   This is the default RETEP processing mode.
3 use mode 2 then search for an isolated LEAVE instruction near the end of the traced routine.
4 use mode 2 then search for an isolated RET instruction near the end of the traced routine.
5 use modes 2 and 3 combined.

Note: Version 2.0 of TRCUST only uses this search mode but is less reliable than mode 2 alone.

Warning:

Use of RETEP can run the risk of having a tracepoint not generated on an instruction boundary. The result of which would be unpredictable and probably cause the traced program to trap. This risk is greatly reduced by allowing the RETEP mode to default to /RM2. If the user chooses to widen the RETEP criteria then appropriate validation of the resulting tracepoints must be made.

6 use mode 4 then search for an isolated JMP instruction near the end of the traced routine.
7 use mode 6 but search for an isolated LEAVE instruction near the end of the traced routine, before searching for JMP.

Note:

Successful location of return tracepoints can only be guaranteed with unoptimised IBM C and C6 CodeView information where the return point is given explicitly. More advanced compilers do not necessarily generate a single return point. They may even generate common epilog code for multiple routines hence the following limitations should be noted:

Only the last return from a routine is located.

Only mode 1 can accurately determine the last return, however this is only available to code-view version 0 modules.

Searching for instruction sequences may result in a tracepoint erroneously being placed within an instruction.

It may not be possible to use RETEP to define the return tracepoint for modules that use private (particularly optimised) calling conventions.
The default mode is 2. This reasonably safe. Higher modes are less safe.

Use mode 5 for compatible behaviour with earlier versions of TRCUST. However, note that earlier versions of TRCUST erroneously permit certain tracepoints which are not permitted using mode 5.

To avoid possible errors, JMP instructions with opcode 0xff are not selected in modes 6 and 7.

For an alternative method of specifying return tracepoints refer to the TP keyword of the TSF TRACE statement and how to use source line number references.

//RP (optional)

modifies RETEP processing to allow Pascal Return instructions (RET n) to be included in the search criteria for return tracepoint location. Use this with 16-bit code where you know that pascal returns are generated.

//RSnn (optional)

specifies how far RETEP will search from the end of a routine to find the return instruction sequence. This defaults to 18 bytes however RETEP will not search before the start of the routine.

//Wh (optional)

is the level of error messages to be displayed, where n can be 0, 1, or 2. The possible message levels are shown below along with the messages that each displays:

0 fatal and severe messages
1 fatal, severe, and error messages
2 all (fatal, severe, error, and warning) messages

A message level of 2 is the default.

Symbolic Debug Support

Source Level Symbolic Support

MAP File Support

Building a Module

Source Level Symbolic Support

If the module has been compiled and linked with the debug options:

1. /Ti on the C/Set2 or VisualAge language compiler for C source files
2. /CO or DE on the link command

then the Trace Customizer can look into the module to extract symbolic or debugging information. In this case addresses may be specified symbolically. Having generated trace definitions there is no further use for symbolic information. Thus debugging information may be safely removed by using the DEBDEL utility.

Notes:

Not all source files must be C language when using symbolic support. Assembler and other languages may be used as long as they generate Microsoft 16-bit CodeView or IBM HLL version 3 and 4 symbolic debugging information.

Tracepoints may be defined by filename and line number references or by symbolic names.

Tracepoints may reference local or global variables.
If symbolic names are referenced then the following points should be observed:

Some compilers will modify external or public symbolic names used in the program. If this is done then the modified form must be used when referenced in the TSF. The modification rules are compiler dependent however three common cases are cited below.

Names must be used case sensitively.

With IBM C/2 and Microsoft C 6.0, external routine names have an underscore prefix to their names when declared using the cdecl convention and are capitalised when declared using the Pascal convention. Symbols for routine names treated this way refer to the first instruction of the routine. When debugging information is present, the unaltered name is also usable and refers to the first instruction following the prologue code, that is creation of the entry stack frame. For example both _main and main may be used. The former refers to the true entry point and the latter after the prologue code. The advantage of using the latter form (main) is that symbolic references may be made to the parameters passed; in this example argc argv and envp.

C runtime library routine names are usually prefixed with an underscore.

With IBM CSET/2 and IBM VisualAge C symbolic names defined by the user are not prefixed with an underscore. However C runtime library routine names are usually prefixed with an underscore. Symbols for routine names defined by the user refer to the initial instruction of the routine when compiled without symbolic debugging information and to the first instruction following the prologue code when debugging information is present. In the latter case, symbolic references may be made to parameters passed to the routine. In the former case this is not possible except by making EBP relative references and using the /PREINV command line switch - see MAP File Support for more information in using symbolic references without the presence of symbolic debugging information.

With IBM VisualAge C++ routine names are subject to name mangling where class-hierarchical information is encoded into the externalised name. The mangled form of the name must be used in TSF references.

In all cases symbolic name lengths are restricted to a maximum length of 255 characters.

--------------------------------------------

MAP File Support

The Trace Customizer can also use the symbolic information in the MAP file produced by the linker. All public symbols will be listed with their offsets in the module being traced. This is not as complete a support as offered by the debug compile option for C language source files, but it does allow entry points, public labels and global data to be referenced symbolically within the TSF. Note that the use of a MAP file is NOT language dependent.

Notes:

When using a MAP file, if the symbolic name is a C language entry point, it will be case sensitive (unless the /I command line switch has been specified).

Some compilers will modify external or public symbolic names used in the program when they appear in the MAP file. If this is done then the modified form must be used when referenced in the TSF. The modification rules are compiler dependent however three common cases follow:

With IBM C/2 and Microsoft C 6.0, external routine names have an underscore prefix to their names when declared using the cdecl convention and are capitalised when declared using the Pascal convention. Symbols for routine names treated this way refer to the first instruction of the routine.

With IBM CSET/2 and IBM VisualAge C symbolic names defined by the user are not prefixed with an underscore. MAP file symbols for routine names defined by the user refer to the initial instruction of the routine, regardless of the presence of debugging information, unless the /PREINV command line switch is specified. When /PREINV is used TRCUST will search for the first instruction following the prologue code, that is, after the entry stack frame has been created.

With IBM VisualAge C++ routine names are subject to name mangling where class-hierarchical information is encoded into the externalised name. The mangled form of the name must be used in TSF references.

In all cases symbolic name lengths are restricted to a maximum length of 255 characters.

C runtime library routine names are usually prefixed with an underscore.
Building a Module

To trace only public procedures, you only need your MAP file that was generated by linking your module.

To trace local variables in C language routines, compile the C programs with the debug option and assemble the ASM routines with public symbols. Link all the OBJs together with the debug option (/CO) and run TRCUST on the executable module. You can now strip the debug information from the executable file by either relinking the OBJs without the debug option or by using the DEBDEL tool to delete the debug information from the executable module file.

TDF and TFF File Usage

The TDF, and TFF files produced by TRCUST are used in the following way:

```
.TSF
V
TRCUST Util

V
.TDF
.TFF
V
TRACE Util
V
tracepoints set
V
tracepoint hit
V
OS/2 kernel
V

V
trace buffer
> TRACEFMT Util
V
formatted
trace records
```

How TRCUST fits into the system

To trace a module do the following:
1. Define the tracepoints and data to be traced in the TSF.

2. Invoke the Trace Customizer using the TSF as input.
   
   This produces two files, a TDF and a TFF.

3. Put the TDF file in the same directory the module to trace resides, put the TFF file in a directory accessible by TRACEFMT. It is suggested that all TFF files reside in the same subdirectory, an example directory would be `\OS2\SYSTEM\TRACE`, which is the default directory searched by TRACEFMT. However a private directory may be specified by using the `/t` command line option or the `Set TFF Path` menu pulldown of TRACEFMT.

4. Invoke the OS/2 TRACE command using either the name of the TDF or the module name instead of the major code value, which is done only with static trace.
   
   **Note:**
   
   The TRACE command will assume that a simple unqualified name refers to a TDF and will search, `\OS2\SYSTEM\TRACE`, the current directory and DPATH for the TDF. If it is still unable to locate the TDF then the name is assumed to be a DLL and LIBPATH is searched for the TDF.
   
   If a qualified name is used then TRACE will assume this to be the traced module and will attempt to locate the TDF in the module's directory, by searching the current directory then LIBPATH for DLLs or PATH for non-DLLs.
   
   This activates the tracepoints, causing the trace data to be saved in the system trace buffer.

5. The OS/2 TRACE command can be used to turn tracing off at any time.

6. The system trace buffer may be captured for formatting at a later time using either of the TRACEGET or TRSPOOL utilities. Or alternatively TRACEFMT may be use to capture and format the trace buffer immediately.
   
   TRACEFMT uses the major code to determine the TFF file and uses the formatting string corresponding to the minor code value to format the data in the RAS trace buffer and output it to the screen, file or printer.
   
   Tracepoints may selectively be turned on or off at any time, by reference to individual minor codes or a group or group+type combination.
   
   Active tracepoints may be queried using either:
   
   ```
   TRACE /Q
   ```
   
   or
   
   ```
   DTRACE QUERY /X /A
   ```
   
   The latter example gives detailed information on tracepoint status and location within each module.

--------------------------------------------

Symbols and Abbreviations Used in the Document

- `[...]` denotes optional items.
- `[..., | ... | ...]` denotes a list of optional items, zero or more of which may be chosen.
- `[, ..., | ...]` denotes a list of items of which ONE must be chosen.
- `item...` denotes that `item` is repeated zero or more times.
- `statement...` denotes this example is incomplete.
- `nnn` is a number in the range 0-255 inclusive.
- `nmm` is a number in the range 0-65535 inclusive.

All numbers and values can be entered in decimal form or in C hexadecimal form (0x...).
Trace Source File

This section details the statements that can appear within a trace source file. Examples are given of TRACE statements.

TSF Format

The layout of a trace source file is:

```
Header
Type List Definition
  (optional)
Group List Definition
  (optional)
Tracepoint Definitions
```

Note:

- Comments may be freely inserted anywhere in the trace source file. A comment is identified by a ; or by using C syntax comments anywhere in the file. A C comment has start and end delimiters, namely /* and */. C type comments may span lines, and may be nested.

- Below are sample TSF files. See Sample Trace Source Files for more examples.

```plaintext
; Sample trace source file depicting dynamic tracing for OS calls compiled
; with 32-bit addressing
MODNAME = doscall1.dll
MAJOR  = 100 /* this is decimal, would be 0x64 if specified hex */
MAXDATALENGTH = 200 /* max bytes logged per tracepoint is 200 */

TYPELIST NAME=PRE, ID=1,
NAME=SYS, ID=0x40,
NAME=API, ID=128,
NAME=POST, ID=0x8000
  /* decimal, if hex would be 0x80 */

GROUPLIST NAME=MEM, ID=2,
NAME=PS, ID=0x5,
NAME=MOS, ID=0x13,
NAME=DOS, ID=0x2B
  /* would be 43 if decimal */

TRACE MINOR=0x0001,
  TP=.DosOpen,
  TYPE=(PRE, API),
  GROUP=DOS,
  DESC=":(OS) DosOpen  Pre-Invocation",
  FMT="Major = \%X Minor = \%Y",
```
FMT="            EAX = %D",
FMT="            FileName = %P%S",
REGS=(EAX),
ASCIIZ32=(.FileName,DIRECT,128)

TRACE MINOR=0x7001,  /* Puts tracept on code at line 28 */
    /* of file dosopenl.c. Debug */
    /* info is needed to use this. */
    TP=@dosopenl.c,28,
    TYPE=(API),
    GROUP=DOS,
    DESC="(OS) CheckParm After Createhandle",
    FMT=" New handle = %P%W",
    MEM32=(.handle,DIRECT,2)

TRACE MINOR=0x8001,  /* Post-invocation tracing at */
    /* procedure DosOpenC return point. */
    TYPE=(API,POST),
    GROUP=DOS,
    DESC="(OS) DosOpenC Post-Invocation",
    FMT=" Return Code = %P%W",
    FMT=" Variable Rec= %P%U",
    MEM32=(.retcode,DIRECT,2),
    MEM32=(.var_struct,DIRECT,LEN)

------------------------------

TSF Header

This defines common information for the module to be traced. The format is:

```
[ MODNAME = [d:][path]Name ]
[ MAJOR = nnnn ]
[ MAXDATALENGTH = nnnn ]
[ TDFID = nn ]
```

where:

MODNAME=[d:][path]Name where:

- **d:**
  is the drive containing the module. If not specified the current drive is used.

- **path**
  is the path to the module. If not specified the current path is used.

- **Name**
  is the name of the executable module to be traced. If an extension is not specified and the Name is not OS2KRNL, an extension of DLL is appended to Name.

Path information may be specified but is normally discarded before storing module name information in the TDF unless the /P switch is specified on the command line. TRCUST uses the path information to locate and read the load module while processing the TSF. If either the name or path differs from the module's installed name or path then the /L= command line option may be use to direct TRCUST on while module file name to read.

**Note:** MODNAME is required only for dynamic tracepoint definitions with TRCUST version 2.22 and later.

MAJOR=nnnnn (optional)
defines the major trace ID allocated to this module. It may be in the range 1 to 65535 decimal or specified 0x1 to 0xffff hex. The
default value is 1. The major trace ID is part of the data placed in the trace buffer when a tracepoint is executed.

Only one major code is permitted per module.

Notes:

Prior to fix pack 35 for OS/2 Warp V3.0 and for OS/2 Warp V4.0 only major codes 0x1 - 0xff may be used. After fix pack
35 for OS/2 Warp V3.0 the major code range of 0x1 - 0xffff permissible. TRCUST version 2.2 or later is required if the
extended range of major codes is to be used.

Also note that the default major code (1) is reserved by OS2 for dekko tracepoints. Ranges available to the user are:
245-255 and 0x8000-0xffff.

Major code 0 is permissible but only with the \RAS switch specified on the command line. However this is reserved for
specific use by system use and should not normally be used.

MAXDATALENGTH=nnnn (optional)

defines the maximum amount of data that a single tracepoint call will insert into the trace buffer.

The length may be in the range 20 to 4099 decimal or specified 0x14 to 0x1003 hex. The default value is 512.

Note:

Prior to fix pack 35 for OS/2 Warp V3.0 and fix pack 10 for OS/2 Warp V4.0 MAXDATALENTH defaults to 128 and may
only specify a value in the range of 20 to 512.

TDFID=nn (optional)

defines an identifier in the range 0 to 65565 for use with the GETVARS and QUERY functions of the DTRACE command. The
default TDFID is 0 and is normally allowed to default.

--------------------------------------------

Typelist Definition

This defines the optional typelist event IDs. For more description and examples of event types see the online help for the trace command.

The intent of type definitions is to assign one or more categories to a tracepoint so that they may be selected as a subset of one or more
groups by the TRACE command. For example, tracepoints at routine entry points might be assigned a type of PRE, while those defined at
return points might be assigned a type of POST. Each tracepoint may be associated with 0 or more types.

The format is:

TYPELIST NAME=TypeName,ID=TypeValue,
[NAME=TypeName,ID=TypeValue,]...

where:

NAME=Typename

defines a 1-8 byte character string used to reference the TypeValue in the tracepoint definitions. All TypeNames and GroupNames
within a TSF must be unique.

ID=TypeValue

defines a bit value of the form 2**y where y in range 0 to 15, permitting a maximum of 16 types to be defined in a single TSF. This
can be decimal or specified 0xnnnn for hex.

An example TYPELIST definition follows:

TYPELIST NAME=PRE,ID=1,
NAME=SYS,ID=0x40,
NAME=API,ID=128,
NAME=POST,ID=0x8000,.....
Grouplist Definition

This defines the optional grouplist IDs. For more description and examples of groups see the online help for the trace command.

The intent of group definitions is to assign subsets of tracepoints to a meaningful category so that they may be selected by group name using the TRACE command. For example, tracepoints associated with file system APIs might be assigned to the FS group. Each tracepoint may be associated with at most one group. Note the difference between group and type association in this aspect.

The format is:

    GROUPLIST NAME=GroupName,ID=GroupValue,

    [NAME=GroupName,ID=GroupValue,]...

where:

NAME=GroupName

defines a 1-8 byte character string used to reference the GroupValue in the tracepoint definitions. There are a maximum of 48 GroupNames allowed in a TSF file. All TypeNames and GroupNames within a TSF must be unique.

ID=GroupValue

defines a word value in the range 1 to 65535 decimal or 0x1 to 0xFFFF hex.

An example GROUPLIST definition follows:

    GROUPLIST NAME=MEM,ID=2,
    NAME=FS,ID=0x5,
    NAME=MOU,ID=13,.....

Tracepoint Definitions

The tracepoint address and the data to be traced are specified by the TRACE statement. There are a maximum of 65535 tracepoints permitted in a trace source file.

The format of the TRACE statement is:

    TRACE [MINOR=minorcode,]
    TP={@STATIC,[@filename,linenum,].name[{-|}offs][,RETEP=[retopt[{-retopt...}]]],
    [OPCODE=0xnn,]
    [TYPE=(typename[,typename...]),
    [GROUP=groupnam,]
    [DESC="Tracepoint description",]
    [FMT="Formatting string",]...
    [LEN=(length_spec,flag),]
    [DATA_STMT,]...

The TRACE keyword delimits a tracepoint definition statement. The definition is considered complete when the next TRACE keyword is encountered or the end of file is reached. There is one TRACE statement for each tracepoint.

LEN is used to log variable length records. A DATA_STMT must immediately follow the LEN statement. LEN will give the location of a one word field containing the number of bytes to log for the following DATA_STMT.
MINOR Keyword

The MINOR parameter is an optional keyword parameter. If it is specified in the first tracepoint definition, it must be specified in every tracepoint definition. If it is not specified in the first tracepoint definition, it cannot be specified in any of the subsequent tracepoint definitions. It should be coded as:

MINOR=nnnnn,

where:

nnnn

is a decimal number from 1 to 65535 or a hex number from 0x1 to 0xFFFF. This represents the minor code for the tracepoint, which should normally be unique within the major code specified for this module. When duplicates are encountered, the original trace definition is saved and duplicates are discarded with an error message.

Duplicate minor codes within major code are permitted when the /D command line switch is specified. However there are precautions to note:

Any formatting information specified in trace definitions with duplicate minor codes is discarded. Only the original is retained in the generated TFF.

Use of duplicate minor codes should be limited to tracepoints where a common formatting template may be used. For example, multiple exit points from a routine.

If minor codes are not specified at all in the TSF then TRCUST will generate them sequentially for each trace definition encountered starting with 1. However, if any definition has an explicit minor code specified then all definitions must have explicit minor codes specified.

TP Keyword

The TP parameter is a required keyword parameter. If TP is specified more than once for a single tracepoint definition, the tracepoint is discarded. TP has three mutually exclusive definitions which can be coded as:

TP=@STATIC,

where:

STATIC

defines this tracepoint entry to be used only for creating a trace format statement for the TFF file. No tracepoint definition is created for the TDF, and the only other TRACE parameters that will be used are DESC, MINOR and FMT. This is used to create trace formatting information for static tracepoints (or dynamic tracepoints generated directly by the DTRACE command). If the TSF contains only @STATIC directives, no TDF files are created and MODNAME is then not required.

TP=@filename,linenum,

where:

filename

is an ASCII string specifying the name (including extension) of a source filename used in creating the module. The source filename is stored in the debug information contained in the executable module, so debug information must exist to use this parameter. The
filename is not case sensitive.

**Note:**

path information may be specified with *filename* when also specified at compilation time.

**linenum**

is a decimal number specifying the line number in the given source file name to place the tracepoint.

**Note:** Debug information must exist to use this option. The statement at the given source line number may have been rearranged during compiler optimization, so the developer must use this with caution. If the line number is not found in the debug information, the tracepoint is applied at the next line number defined in the debug information and a warning message is issued to the user.

An example to apply a tracepoint to line 35 of file stubfile.c is:

```
TRACE MINOR=0x700A,              /* puts tracepoint on code at line */
      TP=@stubfile.c,35,.....    /* 35 of source file stubfile.c */
```

Use of line number reference affords great flexibility in tracepoint location, however the user will need to modify the TSF file every time referenced source files are updated. To facilitate this the **MAKETSF** utility may be used to generate line number references automatically from TSF statements embedded as comments in C or Assembler source.

```
TP=.name[+|-]offs[,RETEP[=retopt[+retopt...]]],
```

where:

**name**

is a public label or an entry point name of a procedure to be traced. The "." preceding *name* is required. *Name* must be found in the debug information in the module or *name* must be a public symbol as found in the MAP file. If debug information is used, the address of this tracepoint will be immediately following the prologue of the procedure. If MAP information is used, this address points to the opcode at the given label unless the /PREINV command line switch is specified. In this case TRCUST attempts to locate the instruction following the

```
MOV EBP,ESP
```

or

```
MOV BP,SP
```

instructions as with done when debug information is used. Use of /PREINV allows parameter and local variable references to be made relative to EBP in non-optimised code.

See **Source Level Symbolic Support** and **MAP File Support** for information on using symbolic references.

**offs** *(optional)*

is a decimal (specified as nnnnnnnn) or hex (specified as 0xnnnnnnnn) offset from the entry point address.

**RETEP** *(optional)*

specifies that the tracepoint will be inserted at the *return* address corresponding to this entry point.

Default criteria for specifying the return point are specified by the /RM RETEP mode command line option. TRCUST will search back from the end of the routine for the instruction sequence matching the RETEP mode. The length of the search is governed by the /RS command line option. The default RETEP mode may be overridden per tracepoint by coding one or more optional *retopt* keywords, separated by a + sign with RETEP. The following keywords are allowed:

- **CV** use CodeView information.
- **LRET** search for a LEAVE+RET or POP EBP + RET instruction sequences near the end of the routine.
- **RET** search for RET instruction sequences near the end of the routine.
- **JMP** search for JMP instruction sequences near the end of the routine.

**Note:** to reduce the hazard of generating a tracepoint on a non-instruction boundary, JMP instructions with 0xFF opcodes are not selected.
LEAVE

search for a LEAVE instruction near the end of the routine.

Note:

If more than one option is specified the RETEP uses the following order of precedence: CV, LRET, RET, LEAVE, JMP.

When the RETEP is used, the name must be a valid entry point to a procedure.

The RETEP option depends upon the manner in which a C compiler generates its code. Therefore this option may not work will some of the newer compilers or with code optimisation.

Warning:

Use of RETEP can run the risk of having a tracepoint not generated on an instruction boundary. The result of which would be unpredictable and probably cause the traced program to trap. This risk is greatly reduced by allowing the RETEP mode to default to /RM2. If the user chooses to widen the RETEP criteria then appropriate validation of the resulting tracepoints must be made.

Note: For ASM functions to accomplish tracing, a label must be made public to have a tracepoint applied. Therefore, to accomplish "POST" tracing, a label must be made public at the return statement.

Partial examples of Pre/Post tracing of DosOpen follows:

```
TRACE  MINOR=0x0001,
       TP=.DosOpen,.....           /* Pre-invocation tracing */
TRACE  MINOR=0x8001,
       TP=.DosOpen,RETEP,.....    /* Post-invocation tracing */
```

Note: It is not possible to set dynamic tracepoints on the following machine instructions:

- 0x9C  PUSHF
- 0xCC  INT 3
- 0xCD  INT n
- 0xCE  INTO
- 0x62  BOUND
- 0x69  IMUL
- 0x6B  IMUL
- 0xF6  DIV, IDIV, MUL, IMUL
- 0xF7  DIV, IDIV, MUL, IMUL

TRCUST gives an error for these opcodes and the tracepoint is rejected.

Note: No more than one tracepoint may be applied to a given address.

--------------------------------------------

OPCODE Keyword

The OPCODE parameter is an optional keyword parameter.

```
OPCODE=0xnn,
```

where:

nn

is the expected one byte hex opcode to be found at the tracepoint address and TRCUST verifies the value with that in the module. The opcode of the instruction being traced must be the same as this value or an error message is issued and the tracepoint is rejected. This may be used to verify the opcode expected at the address specified by the TP parameter. This may be useful when using TP = @filename,linenum to ensure the requested instruction is traced.
TYPE Keyword

The TYPE parameter is an optional keyword parameter that defines the event types of this tracepoint. For more description and examples of event types see the online help for the trace command. Also see the TYPELIST statement for information on how types are used.

```
TYPE={typename[,typename...]},
```

where:

*typename*

is an ASCII string specifying the type of this tracepoint. The typename symbol must have been previously defined by the TYPELIST statement. If an invalid typename is given, the tracepoint will be discarded and a message issued.

The final type value is obtained by logically combining each type name value using the OR operator. If TYPE is omitted, the trace statement will have a typevalue of 0.

---------------------------------------------

GROUP Keyword

The GROUP parameter is an optional keyword parameter that defines the group this tracepoint belongs to. For more description and examples of groups see the online help for the trace command. Also see the GROUPLIST statement for information on how types are used.

```
GROUP=groupnam,
```

where:

*groupnam*

is an ASCII string specifying which group this tracepoint belongs. The groupname symbol must have been previously defined by the GROUPLIST statement. If an invalid groupname is given, the tracepoint will be discarded and a message issued.

If GROUP is omitted, the trace statement will have a groupvalue of 0.

---------------------------------------------

DESC Keyword

The DESC parameter is used to produce a description for the tracepoint that is output as the first line of formatted data. It should include the entry point name of the procedure being traced and whether this is an entry or return point. The descriptive string is enclosed in double quotes as for a C language string. The DESC parameter is required if any FMT specifications are present.

The recommended formats for such strings are as follows:

```
DESC="name Pre-Invocation",
DESC="name Post-Invocation",
```

where:

*name*
is the system component (in parentheses) followed by the entry point name of the procedure.

Pre-Invocation

identifies this tracepoint as an entry point, that is, before the function has been executed.

Post-Invocation

identifies this tracepoint as a return point from the function.

The words Pre-Invocation and Post-Invocation are not mandatory, merely recommendations to be compatible with the base OS/2 tracepoints, when formatted. If a tracepoint is inserted in the middle of a procedure it will be appropriate to use different wording. The Trace Customizer does not check the wording.

An example of pre-invocation and post-invocation tracepoints follow:

```
TRACE  MINOR=0x0001,
   TP=.DosOpen,
   DESC="(OS) DosOpen Pre-Invocation",.....

TRACE  MINOR=0x8001,
   TP=.DosOpen,RETEP,
   DESC="(OS) DosOpen Post-Invocation",.....
```

--------------------------------------------

FMT Keyword

The optional FMT parameter is used to produce the formatting string for the trace data. The developer should use these to control formatting the output produced by the Trace Formatter. Each FMT keyword causes CR/LF to be appended to the format string. The formatting string is similar to a C library printf string specification. It consists of ASCII characters and formatting controls enclosed in double quotes as for a C language string. Each formatting primitive describes the format of the data in the trace buffer at the formatting position and must match the data stored in the trace buffer by the data statements described later. See Formatting Trace Data for a description of how the data is stored in the trace buffer and subsequently formatted.

The formatting controls are as follows:

%Innn  Ignore nnn number of bytes in the trace buffer.

This tells the Trace Formatter to skip over the next nnn bytes in the current trace record. This could be used, for example, to skip over unimportant data, traced as a block, and only output the data of interest.

When using this control, nnn represents an ASCII decimal number and must be followed by a space.

statement: FMT = "ignore ten bytes %I10 here",
FMT = "     and two more %I2 here",
generates: ignore ten bytes here
            and two more here

%P  Process the data prefix bytes associated with the trace data.

This tells the Trace Formatter that the next bytes in the trace record are the prefix or header bytes for data logged by the dynamic tracing mechanism. This is required to precede any format control describing data logged from memory. Do not use this before data that was logged from a register and never use with static tracepoints.

%P may be used in combination with any of the following formatting controls: %A, %B, %C, %D, %F, %Q, %S, %U, %W.

%P%x may be specified as %Px.

See Formatting Trace Data for a description of how the data is stored in the trace buffer and the use of this control.

statement: FMT="memory byte = %P%B",
generates: memory byte = C2

%B Output a byte of data in hexadecimal.

statement: FMT = "memory byte = %P%B"
generates: memory byte = 01

%C Output an ASCII character

statement: FMT = "memory bytes = %C%C%C"
generates: memory byte = ABC

%C formats a byte of data as a single ASCII character, with no separating spaces. Bytes outside the range 0x20 - 0x7f are formatted as periods.

Note:
The %C formatting control is only available with TRCUST 2.26 or later and TRACEFMT version 2.2 or later.

%W Output a word of data.

statement: FMT = "register word = %W"
generates: register word = 0001

statement: FMT = "memory word = %P%W"
generates: memory word = 0001

%D Output a double word of data.

statement: FMT = "double word EAX = %D"
generates: double word EAX = 0000 4B2C

statement: FMT = "double memory word = %P%D"
generates: double memory word = 0000 4B2C

%F Output a Flat (0:32 bit) address.

statement: FMT = "flat address EAX = %F"
generates: flat address EAX = 00004B2C

%Q Output a quad word of data.

statement: FMT = "quad word from regs EAX and EBX = %Q"
generates: quad word from regs EAX and EBX = 00004B2C 00000001
%A  Output a segmented (16:16 bit) address.

    statement: FMT = "segmented address in SS:SP = %A"
    generates: segmented address in SS:SP = 00B7:0001

    statement: FMT = "segmented address in memory = %P%A"
    generates: segmented address in memory = 00B7:0001

%R  Repeat the following format control for the rest of the memory that was logged.

    The action of %R differs between dynamic and static trace:
    
    For dynamic trace records, %R will process the 3-byte memory prefix to determine the length of data to which repeat
    formatting applies. %R is used in place of the %P control.
    
    For static trace records, %R applies itself to all the remaining data in the trace buffer. No 3-byte memory prefix is
    examined or processed.
    
%R may be used only with the following formatting controls: %A, %B, %C, %D, %F, %Q, %W, in particular it has no meaning with
%S and %U.

    statement: FMT = "log a variable number of words from memory = %R%W"
    generates: log a variable number of words from memory = 0001 0004

%S  Output an ASCIIZ string.

    The prefix formatting control should always precede this for dynamic tracepoints because the data was logged from memory.

    Note: If the tracepoint is static, then %P should not be used because the string is terminated with a null byte.

    statement: FMT = "string = %P%S"
    generates: string = c:\os2\os2.ini

%U  Format the remainder of the trace record as a sequence of bytes in hexadecimal or dump format depending upon the View
    pull-down selection from TRACEFMT.

    This will output the remaining of the traced data, including any prefix bytes.

    statement: FMT = "garbage = %U"
    generates: garbage = [00 00 00 03 c2 c1 c4 ff 04 00 09 c0 18]

With fix pack 35 for OS/2 Warp V3.0, fix pack 10 of OS/2 Warp V4.0 and OS/2 Warp E-Server the TRACEFMT utility permits
unformatted data to be displayed in dump format, where both the hexadecimal and ASCII representations of the data are displayed
together with offsets. ASCII characters outside the range 0x20-0x7f are displayed as '.' for example:

    Data [+0000  04 00 5A 00 00 0C 00 2B-DF F6 FF FF 00 00 01 ..Z....+........ ]
    Data [+0010  00 01 C0 ... ]

%X  Output the major event code.

    statement: FMT = "major code = %X"
generates: major code = 00C2

%Y Output the minor event code.

statement: FMT = "minor code = %Y"
generates: minor code = 0081

Note to CMVC Users: To avoid conflicts with source file control information, all formatting specifications can be in upper or lower case. Also, prefix format specifications may be combined with data format specifications. For example, the following create the same format controls in the TFF:

FMT = "%P%W here"
FMT = "%p%w here"
FMT = " %P %W here"

--------------------------------------------
LEN Keyword

The LEN parameter is an optional keyword parameter that defines the length of the variable length record that will follow in the next MEM or MEM32 statement.

LEN=(length_spec,flag),

where:

length_spec

is an address specification that points to the one word length field of the next memory specification. This format can be symbolic_name+nnnnnnnn where symbolic_name is a symbolic memory location and nnnnnnnn is the offset from that symbolic address. The length_spec can also be Flat Register Form. or Segment Register Form.

flag

is a mandatory parameter that identifies the level of indirection to be used on the length_spec. It is one of:

DIRECT

|INDIRECT|"[+][-]iiiiiii]...

DIRECT implies that the length_spec specifies a memory location that contains the length of the variable length record. INDIRECT means that the length_spec contains an address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets iiii of the value found at the given level of indirection.

The following are example LEN statements followed by the memory statement whose length they describe.

TRACE MINOR=.....,
/* Symbol vrecord is a record whose first field is a one */
/* word value that is the total length of the entire */
/* variable length record. */
LEN=(vrecord,DIRECT),
MEM=(.vrecord,DIRECT,LEN),
/* Symbol vrec_ptr is a pointer to a variable length record */
DATA_STMT

There are three types of data that may be traced as part of the optional DATA_STMT section of the TRACE statement.

- Registers
- Memory
- ASCIIZ strings

More than one keyword is permitted in a tracepoint definition. The order of the statements defines the order in which the data is inserted into the trace buffer.

The combined amount of data to be traced for a single tracepoint cannot exceed MAXDATALENGTH. If TRCUST determines that the maximum data size might be exceeded, a warning message is issued but the tracepoint definition will remain valid.

The keywords for tracing the three types of data are REGS, MEM32, MEM, ASCIIZ32, and ASCIIZ.

The REGS keyword identifies which registers are to be recorded in the trace buffer.

The MEM32 keyword is used to record sections of memory in the trace buffer. Access to this memory location is through 32-bit flat addresses from functions compiled using 32-bit addressing. Several MEM32 parameters may be coded at any one tracepoint if several different memory areas are to be traced.

The MEM keyword is also used to record sections of memory in the trace buffer, but access to this memory is through a segment:offset pair. This is used for functions compiled using 16-bit addressing with segment registers. Several MEM parameters may be coded at any one tracepoint if several different memory areas are to be traced.

The ASCIIZ32 keyword is used to record an ASCIIZ string in the trace buffer. This is a special form of the MEM32 keyword and there may be more than one ASCIIZ32 parameter coded for a single tracepoint.

The ASCIIZ keyword is used to record an ASCIIZ string in the trace buffer. This is a special form of the MEM keyword and there may be more than one ASCIIZ parameter coded for a single tracepoint.

REGS Keyword

This is coded as:
REGS={register[,register]...},

where:

register

is one of the following to support OS/2 versions 1.1 and 1.2:

CS,DS,SS,ES,AX,BX,CX,DX,SP,BP,SI,DI,IP,FLAGS

with the addition of the following to support OS/2 version 2.0:

EAX,EBX,ECX,EDX,ESP,EBP,ESI,EDI,EFLAGS,EIP,FS,GS

or the symbolic name of a C language variable declared with the register storage-class specifier as:

.symbolic_name

The same register may appear multiple times in the register list. It will be traced as many times as it appears. Extended registers (E) are 32 bits and logged as two words. All other registers are 16 bits and logged as one word.

Note: To log a C language variable declared with the register storage class, debug information must exist and the variable name is case sensitive. When formatting the data logged from a register variable, remember that there are no memory prefix bytes put into the log buffer.

Example of the REGS statement follows:

/* Given the following declaration in a C language source file: */
register int ret_code;

/* To log registers AX, CX and the register variable ret_code: */
TRACE MINOR=......
REGS=(AX,CX,.ret_code),
FMT="AX=%W  CX=%W  ret_code=%W"

--------------------------------------------

MEM32 Keyword

This is used to log memory in a function compiled using 32-bit flat addressing and is coded as:

MEM32={address_spec,flag,{length|LEN}),

where:

address_spec

is a flat memory address specification as described in Address Specification.

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

D[IRECT]
I[NDIRECT]["[(+-)iiiiii]]
IS

DIRECT implies that the address specifies a memory location to be saved in the trace buffer.

INDIRECT means that the address contains a flat address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets iiiiiii are added to or subtracted from the value found at the given level of indirection.

IS (Indirect Segmented) means that the address contains a segmented address that is dereferenced to obtain the memory location.
length

is the number of bytes at the memory location to be saved in the trace buffer. If length is too big, a warning message will be given, and length will be set to MAXDATALENGTH. If length is 0 an error message will be given, and this tracepoint will be ignored.

LEN

specifies that this is a variable length record to log and the length was specified by the preceding LEN statement. If there was no preceding LEN statement, this tracepoint is rejected. Either length or LEN must be specified, but not both.

Example of the MEM32 statement follows:

TRACE MINOR=.....
/* To log retcode enter the following: */
MEM32=(.retcode,DIRECT,2),
/* s_ptr is a pointer to a structure, log it for 4 bytes. */
MEM32=(.s_ptr,INDIRECT,4),
/* Field 6 bytes into structure pointed at by s_ptr is a */
/* pointer to a structure, log 8 bytes past begin of struct.*/
MEM32=(.s_ptr,INDIRECT*+6*+8,10), /* logs ten bytes */
/* s_ptr points to a variable length record, second field */
/* is the record length (offset 4 from record begining).*/
LEN=(s_ptr,INDIRECT*+4),
MEM32=(.s_ptr,INDIRECT,LEN)
/* s_end points to the end of same variable length record,*/
/* second field is the record length (offset -6 from */
/* record beginning). */
LEN=(s_end,INDIRECT*-6),
MEM32=(.s_ptr,INDIRECT,LEN)

--------------------------------------------

MEM Keyword

This is used to log memory in a function compiled using 16-bit segment:offset addressing and is coded as:

MEM=(address_spec,flag,(length|LEN)),

where:

address_spec

is a segmented memory address specification as described in Address Specification.

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

DIRECT

|INDIRECT|"[+(+-]iiiiii"

DIRECT implies that the address specifies a memory location to be saved in the trace buffer.

INDIRECT means that the address contains a segmented address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets iiii are added to or subtracted from the value found at the given level of indirection.

IF (Indirect Flat) means that the address contains a flat address that is dereferenced to obtain the memory location.
Only far pointers may be dereferenced when using segmented addressing.

length

is the number of bytes at the memory location to be saved in the trace buffer. If length is too big, a warning message will be given, and length will be set to MAXDATALENGTH. If length is 0 an error message will be given, and this tracepoint will be ignored.

LEN

specifies that this is a variable length record to log and the length was specified by the preceding LEN statement. If there was no preceding LEN statement, this tracepoint is rejected. Either length or LEN must be specified, but not both.

--------------------------------------------

ASCIIZ32 Keyword

This is used to log a string in a function compiled using 32-bit flat addressing and is coded as:

ASCIIZ32=(address_spec,flag,maxlength),

where:

address_spec

is a 0:32 bit flat memory address specification as described in Address Specification.

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

D[IRECT]

I[NDIRECT]["[+-]iiiiii]...

IS

DIRECT implies that the address points to a memory location, the contents of which are to be saved in the trace buffer.

INDIRECT means that the address points to a flat address pointer which is dereferenced to obtain the target location to save in the trace buffer. The optional asterisks denote the level of indirection, one for each level. The indirect offsets iiiiiii are added to or subtracted from the value found at the given level of indirection.

IS (Indirect Segmented) means that the address points to a segmented address pointer which is dereferenced to obtain the target location to save in the trace buffer.

maxlength

is the maximum length of the string that will be saved in the trace buffer. It should be no greater than MAXDATALENGTH. The actual length to be traced will depend on where the zero terminating byte is found.

If maxlength is 0 an error message will be given, and this tracepoint will be ignored.

Note: When using dynamic tracing, the OS/2 kernel does not place the terminating null byte into the trace buffer; therefore the prefix byte must be used by the Trace Formatter to obtain the length of the string.

--------------------------------------------

ASCIIZ Keyword

This is used to log a string in a function compiled using 16-bit segment:offset addressing and is coded as:

ASCIIZ=(address_spec,flag,maxlength),

where:
where:

address_spec

is a segmented memory address specification as described in Address Specification.

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

\[
\begin{align*}
\text{DIRECT} \\
\text{INDIRECT}[\{+|-\}i\ldots]
\end{align*}
\]

DIRECT implies that the address points to a memory location, the contents of which are to be saved in the trace buffer.

INDIRECT means that the address points to a far pointer which is a segmented address that is dereferenced to obtain the target location to save in the trace buffer. The optional asterisks denote the level of indirection, one for each level. The indirect offsets \(i\ldots\) are added to or subtracted from the value found at the given level of indirection.

IF (Indirect Flat) means that the address points to a far pointer which is a flat address that is dereferenced to obtain the target location to save in the trace buffer. Only far pointers may be dereferenced using segmented addresses.

maxlength

is the maximum length of the string that will be saved in the trace buffer. It should be no greater than MAXDATALENGTH. The actual length to be traced will depend on where the zero terminating byte is found.

If \( \text{maxlength} \) is 0 an error message will be given, and this tracepoint will be ignored.

Note: When using dynamic tracing, the OS/2 kernel does not place the terminating null byte into the trace buffer; therefore the prefix byte must be used by the Trace Formatter to obtain the length of the string.

Address Specification

The syntax for specifying a memory address given here applies to the MEM32, MEM, ASCIIZ32 and ASCIIZ keywords above.

An address is specified in one of the following forms:

1. Symbolic name form (can be used for MEM32, MEM, ASCIIZ32, and ASCIIZ).
2. Flat register form (can be used only for MEM32 and ASCIIZ32).
3. Segment register form (can be used only for MEM and ASCIIZ).

Symbolic Name Form

This is coded as:

\[
\text{.name}\{\{+|-\}n\ldots\{+|-\}i\ldots\},
\]

where:

name

is a symbolic name of a memory location. The "." is required before the name. The debug information in the module is checked for
the name and if not found and a MAP was given, the MAP is checked. An error message is output by the Trace Customizer if the symbol is not found and the trace definition is ignored.

The name is normally case sensitive and may be subject to modification by the compiler.

See Source Level Symbolic Support and MAP File Support for information on using symbolic references.

If the \# command line switch is specified then MAP file symbolic references may be may case-insensitively.

name is limited to a maximum length of 255 characters. If a symbol exceeds this length then TRCUST will only use the first 255 characters. A warning message is issued when name truncation occurs.

nnnnnnnn (optional)

is a displacement from the symbolic address. If hex the syntax is 0xnnnnnnnn.

iiiiiiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiiiiiii. This specifies a displacement from the final address when using INDIRECT, IF (Indirect Flat) or IS (Indirect Segmented) addressing.

--------------------------------------------

Flat Register Form

This is coded as:

\[ F_{breg}\{+|-}ireg\}...\{+|-}nnnnnnnn\}...\{+|-}iiiiiiii \],

where:

breg

is a flat model (0:32 bit) base register and is one of:

EAX,EBX,ECX,EDX,ESP,EBP,ESI,EDI,EIP

ireg (optional)

is an extended data, base or index register. More than one ireg may be used to define a displacement from the flat register value to the memory location. It may be one of:

EAX,EBX,ECX,EDX,EBP,ESI,EDI,EIP

nnnnnnnn (optional)

is an optional fixed displacement to be added to the address calculated in the registers. If hex the format is 0xnnnnnnnn.

iiiiiiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiiiiiii. This specifies a displacement from the final address when using INDIRECT or IS (Indirect Segmented) addressing.

This form of address is calculated at run time.

--------------------------------------------

Segment Register Form

This is coded as:
where:

sreg

is a segment register and is one of:

CS,DS,SS,ES,FS,GS

Note: The prefix R implies that the adjoined segment register is to be used with an initial offset of zero. Thus RDS refers to data addressed by DS:0

dreg (optional)

is a data, base or index register. More than one dreg may be used to define a displacement from the segment register value to the memory location. It is one of:

BP,SP,SI,DI,AX,BX,CX,DX,IP

nnnn (optional)

is an optional fixed displacement to be added to the address calculated in the registers. If hex the syntax is 0xnnnn.

iiiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiii. This specifies a displacement from the final address when using INDIRECT or IF (Indirect Flat) addressing.

This form of address is calculated at run time.

--------------------------------------------

Formatting Trace Data

This section gives a brief description of the formatting process as an aid to generating correct formatting strings.

Each trace record stored in the RAS buffer consists of a header followed by a number of variable length trace data records. The header identifies the major and minor code, time stamp, process ID, etc., and the total length of the trace data for that trace record.

Each MEM32, MEM, ASCIIZ32, or ASCIIZ data statement, coded in the trace source file for a tracepoint, produces an associated data record to be stored in the trace buffer. The data records consist of a 3-byte prefix followed by the trace data. This prefix consists of a status byte followed by the length of the data for that statement. The status byte indicates whether valid data has been traced.

Dynamic trace can only trace data that is resident in memory at the time that the tracepoint is executed. Data may not be able to be traced for two reasons: it resides in a page that is currently paged out or the address specified is invalid. This latter case usually occurs due to tracing indirectly via invalid pointer variables. In either of these two cases dynamic trace sets the status byte accordingly and stores the pointer in the place of the wanted data. No more data is attempted to be traced for this invocation of the tracepoint, but tracing will resume the next time this tracepoint is encountered.

Since the position of these prefix bytes, within a trace record, is dependent on the data being traced and the number of MEM32, MEM, ASCIIZ32, or ASCIIZ statements, the Trace Formatter must be told when to expect the prefix in the trace record. This is the purpose of the %P and %R formatting controls. One of these must be coded in the formatting string at every place a data record is expected, (noting the restrictions on the use of %R specified in the description of the FMTs keyword.

The following values are used in the status byte of the prefix:

0 Valid data to log
1 ASCIIZ string to log
-1 Invalid Selector
-2 Selector not Present
-3 Page not Present
If the prefix indicates an error TRACEFMT will interpret the error and format the failing address.

Note: With ASCIIZ and ASCIIZ32 commands, the prefix is used by the TRACEFMT utility to obtain the length of the string since the string is not null terminated.

Sample Trace Source Files

This section gives four sample TSF files. The first is for a module written in a mix of C and MASM and compiled with 16:16 segmented addressing. The second was compiled with 0:32 flat addressing. The third module consists of routines, some which were compiled using 16-bit segmented addressing and some that were compiled using 32-bit flat addressing. The fourth is for monitoring function references in a module.

TSF Using 16-bit Segmented Addressing

; Trace source file for the xxx dynalink. Compiled with 16-bit offsets.
MODNAME=\c\src\xxx.dll
MAJOR=0xC5
MAXDATALEN=200
; We will want to trace up to 200 bytes in any one trace call.

TYPELIST NAME=API,ID=08,
NAME=SYS,ID=04,
NAME=PRE,ID=02,
NAME=POST,ID=64

GROUPLIST NAME=MEM,ID=1,
NAME=FS,ID=3

/* The following tracepoint does not need debug info, only a MAP file is necessary with label xxalloc public in it. The program must be compiled in 16-bit mode because segmented addressing is used (ASCIIZ instead of ASCIIZ32). This logs the word registers AX and BX and the string pointed at by DS:DI for a max of 20 bytes. */

TRACE MINOR=25, TP=xxalloc,
OPCODE=0x8B, /* the opcode is optional */
TYPE=(API,PRE),
GROUP=MEM,
DESC="(OS) xxalloc Pre-Invocation",
FMT=" AX = %W ",
FMT=" upper BX = %B",
FMT=" lower BX = %B",
FMT=" param = %P%S",
REGS=(AX,BX),
ASCIIZ=(RDS+DI,DIRECT,20)

/* This defines a tracepoint at Foo label. The ten words to log are found indirectly through SS:SP. Note that each word needs a format control but since only one memory access was done, one prefix control is needed. */

TRACE MINOR=0xB0, TP=Foo,
TYPE=(SYS),
GROUP=FS,
DESC="(OS) Foo Pre-Invocation",
FMT=" First Five words = %P%W%W%W%W",
FMT=" Three words ignored %I6",
FMT=" Last Two Words = %W%W",
MEM=(RSS+SP,INDIRECT,20)
/* This defines a tracepoint at Goo label. DS:DI points
to a structure whose second field is a pointer to an
ASCIIZ string. The offset from the first field in the
structure is 4 bytes. Max string size to log is 40 bytes. */

TRACE MINOR=0xB1, TP=.Goo,
    TYPE=(SYS),
    GROUP=FS,
    DESC="(OS) Goo Pre-Invocation",
    FMT="Second field in struct points to %P%S",
    ASCIIZ=(RDS+DI+4,INDIRECT,40)

/* This defines a tracepoint at Hoo label. DS:DI points to
memory that contains a pointer to a structure. We want to
log the third field in the structure (offset 6 from begin
ning of structure). */

TRACE MINOR=0xB2, TP=.Hoo,
    TYPE=(SYS),
    GROUP=FS,
    DESC="(OS) Hoo Pre-Invocation",
    FMT="Third field in struct is doubleword = %P%D",
    MEM=(RDS+DI,INDIRECT*+6,4)

/* This defines a tracepoint at Zoo label. DS:DI points to
memory that contains a pointer to end of a structure. We
want to log the last field in the structure (offset -2 from
end of structure). */

TRACE MINOR=0xB3, TP=.Zoo,
    TYPE=(SYS),
    GROUP=FS,
    DESC="(OS) Zoo Pre-Invocation",
    FMT="Last field in struct is word = %P%W",
    MEM=(RDS+DI,INDIRECT*-2,2)

/* This defines a tracepoint at procedure CheckIt. This
is a C routine compiled with debug information. The
data to log is an ASCIIZ string called NameIt. */

TRACE MINOR=0xB3, TP=.CheckIt,
    TYPE=(PRE),
    GROUP=FS,
    DESC="(OS) CheckIt Pre-Invocation",
    FMT="NameIt = %P%S",
    ASCIIZ=(.NameIt,DIRECT,64)

/* This defines a tracepoint at the return point of the
procedure CheckIt, a C routine compiled with debug.
Status_Rec is a record variable. We want to log the
age field (four bytes from the begin of Status_Rec),
the name (six bytes from Status_Rec that points to
an ASCIIZ string), the age of the next Status_Rec
(a pointer to the next Status_Rec is ten bytes from
the begin of Status_Rec, the age is four bytes from
the begin of the next Status_Rec). */

TRACE MINOR=0x80B3, TP=.CheckIt,RETEP,
    TYPE=(POST),
    GROUP=FS,
    DESC="(OS) CheckIt Post-Invocation",
    FMT="Status_Rec.age = %P%W",
    FMT="Status_Rec.name = %P%S",
    FMT="Status_Rec.next->age = %P%W",
    MEM=(.Status_Rec+4,DIRECT,2),
    ASCIIZ=(.Status_Rec+6,INDIRECT,64),
    MEM=(.Status_Rec+10,INDIRECT*+4,2)

/* This defines a tracepoint at line 58 in the source
file check.c Debug info is needed to use this
type of tracepoint. v_ptr is a pointer to a variable
sized record. The length is 4 bytes past the
beginning of the record. Log that record. */
TRACE MINOR=0x71B4, TP=@check.c,58,
TYPE=(SYS),
GROUP=FS,
DESC="(OS) CheckIt before allocation",
FMT=" Variant Record = %P%W%D%U",
LEN=(v_ptr,INDIRECT*+4),
MEM=(.v_ptr,INDIRECT,LEN)

/* This does not define a tracepoint, it only defines a trace formatting string for minor code 181 (B5 hex). */

TRACE MINOR=0xB5, TP=@STATIC,
DESC="(OS) StaticProcedure Pre-Invocation",
FMT="
  DI = %W FLAGS = %W"

/* This defines a tracepoint at routine LookUp, but no data is to be logged, only the DESC will show up in the Trace log when the tracepoint is formatted. */

TRACE MINOR=0xB6, TP=.LookUp,
TYPE=(SYS),
GROUP=FS,
DESC="(APP) LookUp Pre-Invocation",
FMT="
  AX = %W",
  EBX = %F",
  param = %P%S",
REGS=(AX,EBX),
ASCIIZ32=(FESP+ESI,DIRECT,20)

/* This defines a tracepoint at Foo label. The ten words
to log are found indirectly by using EBP with offset EDI. Note that each value logged needs a format control. */

TRACE MINOR=0xD0, TP=.Foo,
   TYPE=(SYS),
   GROUP=FS,
   DESC="(NEW) Foo Pre-Invocation",
   FMT="   First Five words = %P%W%W%W%W%W",
   FMT="   Three words ignored %I6",
   FMT="   Last Two Words = %W%W",
MEM32=(FEBP+EDI,INDIRECT,20)

/* This defines a tracepoint at Goo label. EAX + EDI points to a structure whose second field is a pointer to an ASCII string. The offset from the first field in the structure is 4 bytes. Max string size to log is 40 bytes. */

TRACE MINOR=0xD1, TP=.Goo,
   TYPE=(SYS),
   GROUP=FS,
   DESC="(NEW) Goo Pre-Invocation",
   FMT="   Second field in struct points to %P%S",
ASCIIZ32=(FEAX+EDI+4,INDIRECT,40)

/* This defines a tracepoint at Hoo label. EBP + EDI points to memory that contains a pointer to a structure. We want to log the third field in the structure (offset 6 from begin of structure). */

TRACE MINOR=0xD2, TP=.Hoo,
   TYPE=(SYS),
   GROUP=FS,
   DESC="(NEW) Hoo Pre-Invocation",
   FMT="   Third field in struct is doubleword = %P%D",
MEM32=(FEBP+EDI,INDIRECT*+6,4)

/* This defines a tracepoint at Zoo label. EAX + EDI points to memory that contains a pointer to end of a structure. We want to log the last field in the structure (offset -2 from end of structure). */

TRACE MINOR=0xD3, TP=.Zoo,
   TYPE=(SYS),
   GROUP=FS,
   DESC="(OS) Zoo Pre-Invocation",
   FMT="   Last field in struct is word = %P%W",
MEM=(FEAX+EDI,INDIRECT*-2,2)

/* This defines a tracepoint at procedure CheckIt. This is a C routine compiled with debug information. The data to log is an ASCII string called NameIt. */

TRACE MINOR=0xD3, TP=.CheckIt,
   TYPE=(PRE),
   GROUP=FS,
   DESC="(NEW) CheckIt Pre-Invocation",
   FMT="   NameIt = %P%S",
ASCIIZ32=(.NameIt,DIRECT,64)

/* This defines a tracepoint at the return point of the procedure CheckIt, a C routine compiled with debug. Status_Rec is a record variable. We want to log the age field (four bytes from the begin of Status_Rec) the name (six bytes from Status_Rec that points to an ASCII string) and the age of the next Status_Rec (a pointer to the next Status_Rec is ten bytes from the begin of Status_Rec, the age is four bytes from the begin of the next Status_Rec). */

TRACE MINOR=0x80D3, TP=.CheckIt, RETEP,
   TYPE=(POST),
   GROUP=FS,
   DESC="(NEW) CheckIt Post-Invocation",
   FMT="   Status_Rec.age = %P%W", 
```
FMT=" Status_Rec.name = %P%S",
FMT=" Status_Rec.next->age = %P%W",
MEM32=(.Status_Rec+4,DIRECT,2),
ASCIIZ32=(.Status_Rec+6,INDIRECT,64),
MEM32=(.Status_Rec+10,INDIRECT*+4,2)

/* This does not define a tracepoint, it only defines a
trace formatting string for minor code 223 (DF hex). */
TRACE MINOR=0xDF, TP=@STATIC,
DESC="(NEW) StaticProcedure Pre-Invocation",
FMT=" DI = %W FLAGS = %W"

/* This defines a tracepoint at routine LookUp, but no
data is to be logged, only the DESC will show up
in the Trace log when the tracepoint is formatted.
LookUp is a C language routine not compiled with
debug and not declared with Pascal
calling conventions; the underscore is needed for
this label. */
TRACE MINOR=0xE0, TP=._LookUp,
TYPE=(SYS),
GROUP=FS,
DESC="(NEW) LookUp Pre-Invocation"

--------------------------------------------
TSF Using Mix of 16-bit and 32-bit Addressing

; Trace source file for the MIXED dynalink.
; Parts were compiled with 16-bit compiler, some with 32-bit compiler.
; The developer must know how the parameters being sent in are
; to be addressed, whether they are segmented or flat addresses.
MODNAME=MIXCALLS.DLL
MAJOR=250
MAXDATALEN=200
; We will want to trace up to 200 bytes in any one trace call.
TYPELIST NAME=API,ID=08,
NAME=SYS,ID=04,
NAME=PRE,ID=02,
NAME=POST,ID=64

GROUPLIST NAME=MEM,ID=1,
NAME=FS,ID=3

/* The following tracepoint is for the routine MixStub.
This was compiled using segmented addressing and
one of the parameters to it is a pointer to a control
block called mix_ctrl. This pointer, found at SS:SP,
is a flat address because the routine that sent it was
compiled with the flat addressing specification.
This logs the mix_ctrl block for 6 bytes. */
TRACE MINOR=95, TP=._MixStub,
TYPE=(API,PRE),
GROUP=MEM,
DESC="(OS) MixStub Pre-Invocation",
FMT=" mix_ctrl = %P%W %W %W",
MEM=(RSS+SP,IF,6) /* is an indirect flat address */

/* The following is for the routine FlatStub. This was
```
compiled using 32-bit flat addresses. A parameter to flatstub is a pointer called p_seg_info. This pointer is a segmented address because the routine calling flatstub was compiled using 16-bit segmented addressing. Log where p_seg_info points for 2 bytes. */

```
TRACE MINOR=0xf0, TP=.FlatStub,
TYPE=(SYS),
GROUP=FS,
DESC="(OS) FlatStub Pre-Invocation",
FMT="                     seg_info = %P%W",
MEM32=.p_seg_info,IS,2)   /* value p_seg_info is a 16-bit */
/* segmented address */
```

Trace Customizer Messages

The messages generated by the Trace Customizer are given below. In addition to the message itself, the source line in error is displayed.

Errors in the FATAL and SEVERE category will cause the compiler to abort immediately.

ERROR messages will normally cause a tracepoint definition to be discarded and processing continues with the next definition.

WARNING messages will allow a valid tracepoint definition to be produced, and the results will normally be as expected.

```
TRCUST(n) severity: message_text
```

where:

- **n**
  - is the line number of the tsf in error or (1) if the command syntax is in error.

- **severity**
  - is the message severity and may take one of the following values:
    - FATAL
    - SEVERE
    - ERROR
    - WARNING

- **message_text**
  - is the text of the message.

Fatal Messages

<table>
<thead>
<tr>
<th>Msg No.</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TSF file not specified</td>
</tr>
<tr>
<td>2</td>
<td>file not found or access denied : %s</td>
</tr>
</tbody>
</table>
3 cannot open file : %s
4 error accessing file : %s
5 error writing to file : %s
6 unable to allocate more memory
7 too many tracepoints in file
8 error reading file: %s, Rc = %s
9 changing file pointer for: %s, Rc = %s
10 unknown EXE header type for: %s
11 invalid path specified in combine file
12 max TFF files to combine is 50
13 all TFFs not have same major code, file: %s
14 invalid MAP file extension given in: %s
15 TCF file not specified
16 filename to long: %s
17 token in TSF file exceeds %s bytes
18 invalid page length, %s bytes specified
19 error detected unpacking page
20 module does not contain code - no page map

Severe Messages

<table>
<thead>
<tr>
<th>Msg No.</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>module name not specified</td>
</tr>
<tr>
<td>34</td>
<td>premature end of file encountered</td>
</tr>
<tr>
<td>35</td>
<td>syntax error : missing '%s' before '%s'</td>
</tr>
<tr>
<td>36</td>
<td>new line in literal</td>
</tr>
<tr>
<td>37</td>
<td>NULL in literal</td>
</tr>
<tr>
<td>38</td>
<td>keyword '%s' expected, '%s' found</td>
</tr>
<tr>
<td>39</td>
<td>symbolic info not given for %s</td>
</tr>
<tr>
<td>40</td>
<td>MAJOR redefinition</td>
</tr>
<tr>
<td>41</td>
<td>TDFID redefinition</td>
</tr>
<tr>
<td>42</td>
<td>MAXDATALENGTH redefinition</td>
</tr>
<tr>
<td>43</td>
<td>line too long in input file: %s</td>
</tr>
<tr>
<td>44</td>
<td>Invalid RETEP criteria specified</td>
</tr>
</tbody>
</table>
## Error Messages

<table>
<thead>
<tr>
<th>Msg No.</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>number expected, '%s' found</td>
</tr>
<tr>
<td>66</td>
<td>unexpected: %s, ignored</td>
</tr>
<tr>
<td>67</td>
<td>minor code not specified</td>
</tr>
<tr>
<td>68</td>
<td>minor code out of range</td>
</tr>
<tr>
<td>69</td>
<td>TYPELIST redefinition, ignored</td>
</tr>
<tr>
<td>70</td>
<td>GROUPLIST redefinition, ignored</td>
</tr>
<tr>
<td>71</td>
<td>TP redefinition, tracepoint ignored</td>
</tr>
<tr>
<td>72</td>
<td>MINOR redefinition, tracepoint ignored</td>
</tr>
<tr>
<td>73</td>
<td>OPCODE redefinition, tracepoint ignored</td>
</tr>
<tr>
<td>74</td>
<td>syntax error: missing '%s' before '%s'</td>
</tr>
<tr>
<td>75</td>
<td>opcode: %s out of range</td>
</tr>
<tr>
<td>76</td>
<td>opcode at TP address cannot be traced</td>
</tr>
<tr>
<td>77</td>
<td>opcode mismatch at address to apply TP</td>
</tr>
<tr>
<td>78</td>
<td>register expected, '%s' found</td>
</tr>
<tr>
<td>79</td>
<td>symbol not found: %s</td>
</tr>
<tr>
<td>80</td>
<td>address not found</td>
</tr>
<tr>
<td>81</td>
<td>segment register expected, '%s' found</td>
</tr>
<tr>
<td>82</td>
<td>trace record incomplete, '%s' required</td>
</tr>
<tr>
<td>83</td>
<td>RPN command record exceeds 255 bytes</td>
</tr>
<tr>
<td>84</td>
<td>invalid parameter: '%s', ignored</td>
</tr>
<tr>
<td>85</td>
<td>invalid ID: '%s', ignored</td>
</tr>
<tr>
<td>86</td>
<td>group/type redefinition: '%s', ignored</td>
</tr>
<tr>
<td>87</td>
<td>typeid redefinition: '%s', ignored</td>
</tr>
<tr>
<td>88</td>
<td>groupid redefinition: '%s', ignored</td>
</tr>
<tr>
<td>89</td>
<td>invalid address specified: %s</td>
</tr>
<tr>
<td>90</td>
<td>line number past end of code for file %s</td>
</tr>
<tr>
<td>91</td>
<td>Debug info does not exist for: %s</td>
</tr>
<tr>
<td>92</td>
<td>line number missing or invalid: %s</td>
</tr>
<tr>
<td>93</td>
<td>filename %s not found in Debug info</td>
</tr>
<tr>
<td>94</td>
<td>duplicate minor code = %s, ignored</td>
</tr>
<tr>
<td>95</td>
<td>duplicate minor code = %s in file %s, ignored</td>
</tr>
</tbody>
</table>
variable LEN parameter not preceding
RPN stack limit of 16 exceeded
invalid flat register specified: %s
total FMT format specs above 4096 bytes
zero length specified, tracepoint ignored
ORBIT redefinition, tracepoint ignored
invalid ORBIT value, tracepoint ignored
opcode defined after ABORT
opcode defined after REMOVE
duplicate TP address, ignored
/D not allowed with /C, ignored
unable to locate end of %s
HLL Debug Info not in order - forcing /NODE
Ignoring unsupported line numbers, version %s
Unrecognised line entry type %s. Line numbers ignored

Warning Messages

<table>
<thead>
<tr>
<th>Msg No.</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>MAXDATALENGTH out of range, 512 used</td>
</tr>
<tr>
<td>130</td>
<td>typename unknown: %s, ignored</td>
</tr>
<tr>
<td>131</td>
<td>groupname unknown: %s, ignored</td>
</tr>
<tr>
<td>132</td>
<td>file: %s, extension invalid, using: %s</td>
</tr>
<tr>
<td>133</td>
<td>‘%s’ expected before ‘%s’, one assumed</td>
</tr>
<tr>
<td>134</td>
<td>too many %s, first 16 types, 48 groups used</td>
</tr>
<tr>
<td>135</td>
<td>name too long: %s, first 8 characters used</td>
</tr>
<tr>
<td>136</td>
<td>linenum in file: %s not found, using #%s</td>
</tr>
<tr>
<td>137</td>
<td>bad object number: %s used for file %s</td>
</tr>
<tr>
<td>138</td>
<td>offset %s is invalid for object number %s</td>
</tr>
<tr>
<td>139</td>
<td>page tracepoint to be applied at not valid</td>
</tr>
<tr>
<td>140</td>
<td>MAXDATALENGTH to log could be exceeded</td>
</tr>
<tr>
<td>141</td>
<td>MAJOR out of range, 1 used</td>
</tr>
<tr>
<td>142</td>
<td>TDFID out of range, 0 used</td>
</tr>
<tr>
<td>143</td>
<td>index too large, high word ignored</td>
</tr>
<tr>
<td>144</td>
<td>unable to determine return point for %s</td>
</tr>
</tbody>
</table>
Using the MAKETSF Utility

The purpose of MAKETSF is to extract dynamic trace definitions imbedded in C or ASM source file to which they relate. MAKETSF will also substitute line number information into those trace definitions that are specified by line number and source file reference.

For example:

```
TRACE TP=@myprog.c,1234
```

This specifies a tracepoint at location corresponding to line 1234 in module whose source is "myprog.c".

The problem with this type of specification is that line number reference will need to be updated whenever the source is changed.

MAKETSF allows the trace definitions to be imbedded in the source as comments but in extraction will generate the correct line number information. It does this by detecting the string "TP=@," or "TP=@ " and then inserting the line number specification. If the TP= keyword explicitly specifies an address expression then the trace definition is extracted without modification.

To use line number references modules must be compiled and linked with symbolic debugging information. For CSET2 and VisualAge the compile option is /TI, and the LINK386 option is /DE.

**Note:** Optimised code may not place tracepoints in the desired location.

For example, in a C program

```
/***DT here is the header
 * MODNAME=my.exe
 * MAJOR=256
 */

void myproc(char * parm1) {
 int result;

 .
 .
 .

/***DT tracepoint definition
 * TRACE TP=8,
 * DESC="a tracepoint",
 * MEM32=(.parm1,I,4),
 * MEM32=(.result,D,4),
 * FMT="parm1=%p%f result=%f"
 */
```

Similarly in an assembler program:

```
;***DT here is the header
 ; MODNAME=my.exe
```
parm1 DD ?

myproc proc
    push ebp
    move ebp, esp
    ...
    ...

;***DT tracepoint definition
;   TRACE TP=0,
;   DESC="a tracepoint",
;   MEM32=(.parm1,D,4),
;   FMT="parm1=%p%f"

Notes:

The comment block must begin in column 1 with either ;***DT or /***DT.
If the C form is used then an * must appear in column 2 of each line. A */ in column 2 ends the comment block.
If the ASM form is used then a ; is required in every column.
If TP=@ is coded then MAPTSF will insert the file name and line number corresponding to the trace definition.

MAKETSF will respond to the $include <filename> directive if coded in column 1. This is intended for use where multiple source modules comprise a single load module. A list of component source files can be coded in a single file using include statements then given to MAKETSF for processing. For example:

A sample include file for MAKETSF

$include myprog.c
$include ..\asm\myproc.asm
$include ..\sub\subr.c  This is a comment

MAKETSF will ignore any line that is not part of a ;***DT or /***DT comment block or a $include statement.
TSF output form MAKETSF is written to the output file if specified, otherwise to STDOUT.
Messages are written to STDERR.

--------------------------------------------

MAKETSF Synatx and command line options

The syntax for MAKETSF is:

MAKETSF <options> input

There are five switch options that MAKETSF supports:

-a
    append output to output file. The default is to replace the output file.
-b<path> specifies the base directory path (or drive) from which input files specifications are assumed to be relative. If not specified then the current directory is assumed to be the base path.

**Note:**

Note: the base path is not included in the file specification of the TP= keyword, whether or not -p is specified.

-n no includes.

This forces MAKETSF to ignore $include directives.

-o<output> specifies the output file. The default is to direct output to STDOUT.

-p include source file path information.

This forces MAKETSF to retain the full file specification from $include statements or the command line input file in the TP= keyword. The default is to strip path information.

-v verbose mode.

This gives information about number of files read and blocks processed.

---------------------------------------------

Using the DEBDEL Utility

The DEBDEL utility may be use to remove debugging information from a load module after TRCUST has generated the TDF. This avoids the need to re-link edit the load module without /DE option.

Removal of debugging information does not affect the validity of the TDF, however if the source is re-compiled then the TDF must be re-built.

The syntax for DEBDEL is:

```
DEBDEL module_file
```

There is only one parameter: **module_file**. This is the load module file to be stripped of debugging debugging information.

**Note:**

DEBDEL updates the module in place without first copying it.

---------------------------------------------

Using the MAPTSF Utility

The MAPTSF utility will generate a TSF from a MAP file. One tracepoint is generated for each public code symbol. Optionally a return tracepoint may be generated for each public code symbol.

The syntax for MAPTSF is as follows:

```
MAPTSF map_file [/MAJOR=major_code] [/MODNAME=name]
```
There is only one required parameter: `map_file`, which specifies the input MAP file. The remaining parameters are optional and have the following meaning:

**/MAJOR**

specifies the major code to be used in the MAJOR= statement of the TSF. If omitted, TRCUST will select the default major code of 1 when compiling the TSF.

**/MODNAME**

specifies the module name to be used in the MODNAME= statement of the TSF. If omitted, MAPTSF will use the module name that appears in the second line of the MAP file. Note, the MAP file excludes the module extension. TRCUST will assume an extension of .DLL if not specified.

**/MAXDATALENGTH**

specifies the MAXDATALENGTH= statement of the TSF. If omitted, TRCUST will assume the default of 512 when compiling the TSF.

**/MINORSTART**

specifies the first minor code. Subsequent tracepoints have incremental minor codes. If omitted, the MINOR= statement is not generated. TRCUST will assume an initial minor code of 1 and increment for each tracepoint.

**/INCLUDE**

specifies a comma delimited list of case insensitive strings used as inclusion criteria for public symbols. An optional trailing * signifies a generic match. If both /INCLUDE and /EXCLUDE are specified then the logical OR of their criteria is used for selection. For example:

```
/incl=dos*,strupr includes all public symbols beginning 'dos' or equal to 'strupr'.
/excl=s*/incl=strupr excludes all public symbols beginning 's' except for 'strupr' and includes everything else.
```

**/EXCLUDE**

specifies a comma delimited list of case insensitive strings used as exclusion criteria for public symbols. An optional trailing * signifies a generic match. If both /INCLUDE and /EXCLUDE are specified then the logical OR of their criteria is used for selection. For example:

```
/excl=*_incl=dos* excludes all public symbols beginning '_' or 'dos'.
/excl=s*/incl=strupr excludes all public symbols beginning 's' except for 'strupr' and includes everything else.
```

**/CASESENSITIVE**

switch applies to /INCLUDE and /EXCLUDE. If specified then the include and exclude strings will be match on a case-sensitive basis.

**/LOGSTACK=n**

specifies the number of bytes of stack to log for entry tracepoints. This causes the following TSF statements to be generated for each entry tracepoint:

```
for 16-bit code -

REGS=(SP,SS),
FMT="Stack pointer SS:SP=%A->",
MEM=(RSS+SP,D,n),
FMT="%W%W"
```

```
for 32-bit code -

REGS=(RIP,SS),
FMT="Instruction pointer RIP:SP=%A->",
MEM=(RSS+SP,D,n),
FMT="%W%W"
```
REGS=(ESP),
FMT="Stack pointer ESP=%F->",
MEM32=(FESP,D,n)
FMT="%R%F"

If /LOGSTACK is specified without a value then 16 bytes is assumed.

/LOGRETURN specifies that for each return tracepoint, the return value in AX/EAX should be logged. This causes the following TSF statements to be generated:

for 16-bit code:

```plaintext
REGS=(AX)
FMT="Returns (ax) %W"
```

for 32-bit code:

```plaintext
REGS=(EAX)
FMT="Returns (eax) %F"
```

/REGISTERS specifies one or more processor registers to be logged, each separated by a comma. The following register mnemonics are supported:

```
AX,BX,CX,DX,CS,DS,ES,FS,GS,IP,SI,DI,SP,BP,FLAGS,
EAX,EBX,ECX,EDX,EIP,ESI,EDI,ESP,EBP,EFLAGS
```

/REGS may be used as a synonym for /REGISTERS.

/RETEP specifies that for each public entry-point in the MAP file, a return tracepoint should be generated using the RETEP parameter of the TRACE statement in the TSF.

/TYPES specifies that generated tracepoints are to be assigned one or more of the following pre-defined types:

- PUB: Public routines - names the begin upper case (ignoring leading underscores)
- PRIV: Private routines - names the begin lower case (ignoring leading underscores)
- PRE: Entry tracepoint.
- POST: Exit tracepoint.

/GROUPS Requests that each of the strings listed be used to define a group. Tracepoints are assigned to a group according to whether a group name matches the beginning of the tracepoint name, ignoring case and leading underscore characters

/TEMPLATE specifies a file where up to four template tracepoint definitions may be specified, one for each of the following categories:

- 16-bit entry points
- 16-bit return points
- 32-bit entry points
- 32-bit return points
The definitions are in a shortened form of the TRCUST TRACE statement syntax. They are appended to each tracepoint of the category to which they apply. All parameters other than MINOR and DESC are permissible. TP and RETEP are specified as follows:

TP=@16 signifies a 16-bit entry-point
TP=@16,RETEP signifies a 16-bit return-point
TP=@32 signifies a 32-bit entry-point
TP=@32,RETEP signifies a 32-bit return-point

Only TP and RETEP may appear on the same line as the TRACE keyword.

For example:

```
TRACE TP=@16
    MEM=(SS:BP+8,I,0x10)
    FMT="16-bytes of parameter 1: %R%W"
```

will append

```
MEM=(SS:BP+8,I,0x10)
FMT="16-bytes of parameter 1: %R%W"
```

to every 16-bit entry tracepoint definition.

--------------------------------------------

System Tracepoints Reference

This chapter documents all static and dynamic tracepoints that are defined in the base OS/2 system code.

Tracepoint definitions are listed in the section: Trace Event Major and Minor Code Assignments. Each tracepoint listing is divided into the following six fields:

- **Event Description**: A description of the event that is being captured (for example, DosOpen Pre-invocation)
- **Tracepoint**: The public symbol or relative location at which the tracepoint is defined.
- **Minor Code**: The minor trace code that is associated with the event.
- **Type**: A qualifier that can be used within the Trace control utility to enable or disable sets of dynamic tracepoints with a single command. **Type Qualifiers** are discussed in more detail in a following section.
- **Group**: A second type of qualifier that can be used within the Trace control utility to enable or disable sets of dynamic tracepoints with a single command. **Group Qualifiers** are discussed in more detail in a following section.
- **Parameters**: The data that is logged by the tracepoint. **Parameter Notation** is discussed in more detail in a following section.

For information on defining tracepoints consult the following:

- The dynamic trace customiser utility **TRCUST**
The DosSysTrace API.

Type Qualifiers

The following Type identifiers are currently defined. They are intended to simplify the identification of components and to aid in selecting events to be traced:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>A pre-invocation event</td>
</tr>
<tr>
<td>POST</td>
<td>A post-invocation event</td>
</tr>
<tr>
<td>API</td>
<td>An external interface</td>
</tr>
<tr>
<td>INT</td>
<td>An internal interface</td>
</tr>
</tbody>
</table>

Typically, pairs of Type identifiers are used (for example, PRE and API).

Group Qualifiers

The following Group identifiers are currently defined. They are intended to simplify the identification of components and to aid in selecting events to be traced:

<table>
<thead>
<tr>
<th>Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXMG</td>
<td>Exception Management</td>
</tr>
<tr>
<td>FS</td>
<td>File System</td>
</tr>
<tr>
<td>IO</td>
<td>Device I/O</td>
</tr>
<tr>
<td>KBD</td>
<td>Keyboard</td>
</tr>
<tr>
<td>LDR</td>
<td>Loader</td>
</tr>
<tr>
<td>LNK</td>
<td>Environment</td>
</tr>
<tr>
<td>LOCK</td>
<td>SMP Spink Locks</td>
</tr>
<tr>
<td>MOU</td>
<td>Mouse</td>
</tr>
<tr>
<td>MSP</td>
<td>Memory Suballocation</td>
</tr>
<tr>
<td>NLS</td>
<td>National Language Services</td>
</tr>
<tr>
<td>PIP</td>
<td>Pipes</td>
</tr>
<tr>
<td>QUE</td>
<td>Queues</td>
</tr>
<tr>
<td>SEL</td>
<td>Selector Management</td>
</tr>
<tr>
<td>SEM</td>
<td>Semaphores</td>
</tr>
<tr>
<td>SIG</td>
<td>Signals</td>
</tr>
<tr>
<td>TIM</td>
<td>Timers</td>
</tr>
<tr>
<td>TK</td>
<td>Tasking</td>
</tr>
<tr>
<td>UT</td>
<td>UnitThunk Processing</td>
</tr>
<tr>
<td>VIO</td>
<td>Video I/O</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Memory Management</td>
</tr>
</tbody>
</table>

Parameter Format Notation

The Parameter portion of a tracepoint listing typically includes both fixed descriptive ASCII strings and variable format descriptors. The following list enumerates the classes of format descriptors that are encountered in the tracepoint listings:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%A</td>
<td>Output a segmented (16:16) address (32 bits)</td>
</tr>
<tr>
<td>%B</td>
<td>Output a byte (8 bits) of data</td>
</tr>
<tr>
<td>%D</td>
<td>Output an ASCII string (n bytes terminated by 00h)</td>
</tr>
<tr>
<td>%F</td>
<td>Output a word (16 bits) of data</td>
</tr>
<tr>
<td>%Q</td>
<td>Output a double word (32 bits) of data</td>
</tr>
<tr>
<td>%W</td>
<td>Output an ASCII string (n bytes terminated by 00h)</td>
</tr>
<tr>
<td>%X</td>
<td>Output a flat (0:32) address (32 bits)</td>
</tr>
<tr>
<td>%Y</td>
<td>Format the remainder of the trace record as a sequence of bytes</td>
</tr>
<tr>
<td>%Z</td>
<td>Output an ASCII string (n bytes terminated by 00h)</td>
</tr>
</tbody>
</table>
DosXxxx API Pre-invocation Tracepoints.

From OS/2 Warp V3.0 fix pack 30 and OS/2 Warp V4.0 fix pack 10 all system API pre-invocation tracepoints have been updated to log the caller's return address. In most cases the meaning of this information is self-evident and the original tracepoint definition has been updated to include a line that displays as follows:

32-bit API:
    Return address = %F
16-bit API:
    Return address = %W:%W

Special consideration needs to be given to APIs involving the KERNEL, SESMGR and QUECALLS. These are APIs the use the Dos prefix in their names and my indirect though DOSCALL1.

The following topics describe each of these cases:

Tracing Kernel API Return Information
Tracing Session Manager API return information
Tracing Queue Calls API return information

Tracing Kernel API Return Information

Kernel APIs

There three schemes that operate when kernel API is called:

Direct call:
    APPL                     > KERNEL
    <

Indirect call, direct return:
    APPL     > DOSCALL1      > KERNEL
    <

Indirect call, indirect return:
    APPL     > DOSCALL1      > KERNEL
    <     DOSCALL1 <

Direct Calls:

Kernel APIs are entered directly from the application. Return information is logged in the following format:
32-bit API:
Return EIP=%F CS=%W

16-bit API:
Return IP=%W CS=%W

CS, EIP and IP refer to the return address in the application.

Use TRACE ON KERNEL(....) to trace these APIs.

Indirect call, direct return:

32-bit Kernel APIs are preprocessed by DOSCALL1 but are returned to directly from the KERNEL. Return information is logged in the following format:

32-bit API:
Return EIP=%F CS=%W, Thunk EIP=%F

Thunk EIP refers to the return address in to the application. Return EIP and CS refer to the return back to DOSCALL1. In most cases the pre-invocation tracepoint is recoded after the kernel has updated the Return EIP with the Thunk EIP value and they will have the same value.

Use TRACE ON KERNEL(....) to trace these APIs.

Indirect call, Indirect return:

32-bit APIs are pre- and post-processed by DOSCALL1. Parameters are logged by the kernel tracepoint, but the return address will only show the direct return back to DOSCALL1. For each API of this form a pre-invocation API in DOSCALL1 has been defined that logs just the return address back to the application. For example, the following shows DosSleep entry to the Kernel, preceded by Dos32Sleep entry to DOSCALL1:

(OS) DosSleep Pre-Invocation
Return EIP=0000C361 CS=DFD7
Timeout Interval = 0000 0000

(OS) Dos32Sleep Pre-Invocation
Return address = 1BDFAA63

For a list of kernel APIs indirected through DOSCALL1 see: Kernel API Tracepoints Indirected Via DOSCALL1.

--------------------------------------------

Tracing Session Manager API return information

SESMGR APIs

SESMGR APIs are entered either directly or indirectly via a SESMGR thunking layer as shown below:

Direct call:

APPL
<

> SESMGR
Thunked call:

\[
\text{APPL} \quad > \quad \text{SESMGR Thunk Layer} \quad > \quad \text{SESMGR} \\
< \quad \text{SESMGR Thunk Layer} <
\]

Direct Calls:

Used for 16-bit APIs. Use TRACE ON SESMGR(....) for tracing APIs and return information.

Thunked Calls:

Used for 32-bit SESMGR APIs. Trace the 16-bit SESMGR API with the 32-bit SESMGR API to record the return information to the original 32-bit caller. For example, to trace the return information and parameters to \textbf{DosStartSession} from a 32-bit caller use: TRACE ON SESMGR(20,34,32788). The following is an example of the traced output:

\begin{verbatim}
(OS) DosStartSession Post-Invocation
New Process Id = 0000
New Session = f0000
Return Code = 0000

(OS) DosStartSession Pre-Invocation
Return address = DFD7:BD0C
Start Data = 0032 0000 0000 0000 62D4 004F 2B54 004F CA10 045F 0000 0000 0000 0000 0001 0000
0000 0000 0000 0000 0000 0000 0000

(OS) Dos32StartSession Pre-Invocation
Return address = 0002E7E2
\end{verbatim}

In this example the return address to the 32-bit application is given in minor code 34.

For a list of indirected SESMR APIs see: \textit{Indirected Session Manager API Tracepoints}.

--------------------------------------------

Tracing Queue Calls API return information

\textbf{QUECALLS APIs}

QUECALLS APIs are entered indirectly via DOSCALL1 as shown below:

\[
\text{APPL} \quad > \quad \text{DOSCALL1} \quad > \quad \text{QUECALLS} \\
< \quad \text{DOSCALL1} <
\]

To trace QUECALLS API parameters and results use TRACE ON QUECALLS(....). To trace the API return addresses use TRACE ON DOSCALL1(....).

\textbf{Note:} The QUECALLS entry points in DOSCALL1 have been grouped using group id QUE. All the API return addresses may be traced by specifying TRACE ON DOSCALL1(QUE).
The following is an example of QUECALLS traced output activated using

TRACE ON DOSCALL1(QUE)
TRACE ON QUECALLS

(OS) DosReadQueue Pre-Invocation
Handle=0014 Element Code=0000
Semaphore Handle=0027 75E4 No Wait Flag=00

(OS) Dos16ReadQueue Pre-Invocation
Return address = 000F:47EC

In this example the return address to the 16-bit caller is given in minor code 360 of DOSCALL1.

For a list of indirected QUECALLS tracepoints see: QUECALLS API Tracepoints Indirected Via DOSCALL1.

-----------------------------

Trace Event Major and Minor Code Assignments

This section lists the overall assignment of major codes for the kernel and related subsystems supported by the trace facility.

<table>
<thead>
<tr>
<th>Usage</th>
<th>Major Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Trace internally generated events</td>
<td>00</td>
</tr>
<tr>
<td>Dekko performance tracepoints</td>
<td>01</td>
</tr>
<tr>
<td>Reserved</td>
<td>02</td>
</tr>
<tr>
<td>Machine Exceptions</td>
<td>03</td>
</tr>
<tr>
<td>Hardware Interrupts</td>
<td>04</td>
</tr>
<tr>
<td>Kernel Services (KERNEL)</td>
<td>05</td>
</tr>
<tr>
<td>Device Helper Services</td>
<td>06</td>
</tr>
<tr>
<td>Miscellaneous Kernel Events</td>
<td>07</td>
</tr>
<tr>
<td>Resource Manager Events (RESOURCE.SYS)</td>
<td>08</td>
</tr>
<tr>
<td>Reserved</td>
<td>09 - 15</td>
</tr>
<tr>
<td>Kernel Services (DOSCALL1.DLL)</td>
<td>16</td>
</tr>
<tr>
<td>Monitor Call Services (MONCALLS.DLL)</td>
<td>16</td>
</tr>
<tr>
<td>Reserved</td>
<td>17 - 21</td>
</tr>
<tr>
<td>Queue Services (QUECALLS.DLL)</td>
<td>22</td>
</tr>
<tr>
<td>Session Manager Services (SESMGR.DLL)</td>
<td>23</td>
</tr>
<tr>
<td>Character I/O Services (OS2CHAR.DLL)</td>
<td>24</td>
</tr>
<tr>
<td>Reserved</td>
<td>25 - 108</td>
</tr>
<tr>
<td>Multi-Media Extensions</td>
<td>109</td>
</tr>
<tr>
<td>Reserved</td>
<td>110 - 137</td>
</tr>
<tr>
<td>Service Name</td>
<td>Starting Address</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>DCAF (Distributed Console Access Facility)</td>
<td>138</td>
</tr>
<tr>
<td>Reserved</td>
<td>139 - 150</td>
</tr>
<tr>
<td>IBM LAN Manager (RTP)</td>
<td>151</td>
</tr>
<tr>
<td>EE LAN Requestor</td>
<td>152 - 159</td>
</tr>
<tr>
<td>Communications Manager APPC</td>
<td>160</td>
</tr>
<tr>
<td>Communications Manager 3270 Emulator</td>
<td>161</td>
</tr>
<tr>
<td>Communications Manager Async</td>
<td>162</td>
</tr>
<tr>
<td>Communications Manager SRPI/DLC</td>
<td>163</td>
</tr>
<tr>
<td>Communications Manager</td>
<td>164 - 167</td>
</tr>
<tr>
<td>Reserved</td>
<td>168</td>
</tr>
<tr>
<td>Communications Manager</td>
<td>169</td>
</tr>
<tr>
<td>Reserved</td>
<td>170 - 171</td>
</tr>
<tr>
<td>Communications Manager</td>
<td>172 - 175</td>
</tr>
<tr>
<td>Database Manager</td>
<td>176 - 191</td>
</tr>
<tr>
<td>PM Shell API (PMSHAPI.DLL)</td>
<td>192</td>
</tr>
<tr>
<td>Reserved</td>
<td>193</td>
</tr>
<tr>
<td>PM Window Management (PMWIN.DLL)</td>
<td>194</td>
</tr>
<tr>
<td>PM Graphics Engine (PMGRE.DLL)</td>
<td>195</td>
</tr>
<tr>
<td>PM Picture Interchange (PMPIC.DLL)</td>
<td>196</td>
</tr>
<tr>
<td>PM Graphics Programming Interfaces (PMGPI.DLL)</td>
<td>197</td>
</tr>
<tr>
<td>PM Print Spooler (PMSPL.DLL)</td>
<td>198</td>
</tr>
<tr>
<td>Reserved</td>
<td>199 - 212</td>
</tr>
<tr>
<td>Transmission Network Manager</td>
<td>212</td>
</tr>
<tr>
<td>Reserved</td>
<td>213</td>
</tr>
<tr>
<td>DBM Archival Logging</td>
<td>222 - 225</td>
</tr>
<tr>
<td>DBM Query Manager</td>
<td>226 - 235</td>
</tr>
<tr>
<td>User Profile Management</td>
<td>236 - 237</td>
</tr>
<tr>
<td>238 - 241</td>
<td>Reserved</td>
</tr>
<tr>
<td>LAN Station Manager</td>
<td>242</td>
</tr>
<tr>
<td>Reserved</td>
<td>243</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>244</td>
</tr>
<tr>
<td>Available for Temporary Application Use</td>
<td>245 - 255</td>
</tr>
<tr>
<td>Kernel Device Manager Internal Tracepoints</td>
<td>256</td>
</tr>
<tr>
<td>Kernel File System Manager Internal Tracepoints</td>
<td>257</td>
</tr>
<tr>
<td>Kernel Virtual DOS Machine Manager Internal Tracepoints</td>
<td>258</td>
</tr>
<tr>
<td>Kernel Module Loader and Manager Internal Tracepoints</td>
<td>259</td>
</tr>
</tbody>
</table>
Trace Event Major Code Assignments

Machine Exceptions Trace Events

The tracepoints for the Machine Exceptions major code are identified in the following table. These tracepoints are static tracepoints. They are compiled with the code.

Trace events for EXCEPT Major Code: 0X0003, sorted by minor code.
Trace events for EXCEPT Major Code: 0X0003, sorted by tracepoint.

Trace Events for EXCEPT Major Code: 0X0003, Sorted by Minor Code
Trace Events for EXCEPT Major Code: 0X0003, Sorted by Tracepoint

(OS) Machine Exception 00 - Divide by 0
(OS) Machine Exception 01 - Single Step
(OS) Machine Exception 02 - NMI
(OS) Machine Exception 03 - Breakpoint
(OS) Machine Exception 04 - Overflow
(OS) Machine Exception 05 - Bounds Check
(OS) Machine Exception 06 - Invalid Opcode
(OS) Machine Exception 07 - Coprocessor Not Available
(OS) Machine Exception 08 - Double Fault
(OS) Machine Exception 09 - Reserved
(OS) Machine Exception 0a - Invalid TSS
(OS) Machine Exception 0b - Segment Not Present
(OS) Machine Exception 0c - Stack Exception
(OS) Machine Exception 0d - General Protection
(OS) Machine Exception 0e - Page Fault

EXCEPT Major Code: 0X0003 Minor code: 1 (0X0001)

Description
(OS) Machine Exception 00 - Divide by 0

Tracepoint
Static tracepoint in EXCEPT.

Minor code
1 (0X0001)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Error Code = %D, CS:EBP = %Q, EFlags = %D
EXCEPT Major Code: 0X0003 Minor code: 2 (0X0002)

**Description**  
(OS) Machine Exception 01 - Single Step

**Tracepoint**  
Static tracepoint in EXCEPT.

**Minor code**  
2 (0X0002)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 3 (0X0003)

**Description**  
(OS) Machine Exception 02 - NMI

**Tracepoint**  
Static tracepoint in EXCEPT.

**Minor code**  
3 (0X0003)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 4 (0X0004)

**Description**  
(OS) Machine Exception 03 - Breakpoint

**Tracepoint**  
Static tracepoint in EXCEPT.
Minor code 4 (0X0004)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----------------------------

EXCEPT Major Code: 0X0003 Minor code: 5 (0X0005)

Description

(OS) Machine Exception 04 - Overflow

Tracepoint

Static tracepoint in EXCEPT.

Minor code 5 (0X0005)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----------------------------

EXCEPT Major Code: 0X0003 Minor code: 6 (0X0006)

Description

(OS) Machine Exception 05 - Bounds Check

Tracepoint

Static tracepoint in EXCEPT.

Minor code 6 (0X0006)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D
EXCEPT Major Code: 0X0003 Minor code: 7 (0X0007)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Machine Exception 06 - Invalid Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in EXCEPT.</td>
</tr>
<tr>
<td>Minor code</td>
<td>7 (0X0007)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Error Code = %D, CS:EIP = %Q, EFlags = %D</td>
</tr>
</tbody>
</table>

EXCEPT Major Code: 0X0003 Minor code: 8 (0X0008)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Machine Exception 07 - Coprocessor Not Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in EXCEPT.</td>
</tr>
<tr>
<td>Minor code</td>
<td>8 (0X0008)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Error Code = %D, CS:EIP = %Q, EFlags = %D</td>
</tr>
</tbody>
</table>

EXCEPT Major Code: 0X0003 Minor code: 9 (0X0009)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Machine Exception 08 - Double Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in EXCEPT.</td>
</tr>
</tbody>
</table>
Minor code  9 (0X0009)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 10 (0X000A)

Description  (OS) Machine Exception 09 - Reserved
Tracepoint  Static tracepoint in EXCEPT.
Minor code  10 (0X000A)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 11 (0X000B)

Description  (OS) Machine Exception 0a - Invalid TSS
Tracepoint  Static tracepoint in EXCEPT.
Minor code  11 (0X000B)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D
EXCEPT Major Code: 0X0003 Minor code: 12 (0X000C)

Description
(OS) Machine Exception 0b - Segment Not Present

Tracepoint
Static tracepoint in EXCEPT.

Minor code
12 (0X000C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 13 (0X000D)

Description
(OS) Machine Exception 0c - Stack Exception

Tracepoint
Static tracepoint in EXCEPT.

Minor code
13 (0X000D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Error Code = %D, CS:EIP = %Q, EFlags = %D

EXCEPT Major Code: 0X0003 Minor code: 14 (0X000E)

Description
(OS) Machine Exception 0d - General Protection

Tracepoint
Static tracepoint in EXCEPT.
**EXCEPT Major Code: 0X0003 Minor code: 15 (0X000F)**

**Description**
(OS) Machine Exception 0e - Page Fault

**Tracepoint**
Static tracepoint in EXCEPT.

**Minor code**
15 (0X000F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Error Code = %D, CS:EIP = %Q, EFlags = %D

Page Fault Linear Address (CR2) = %F

--------------------------------------------

**Hardware Interrupts Trace Events**

The tracepoints for the Hardware Interrupts major code are identified in the following tables. These tracepoints are static tracepoints. They are compiled with the code.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for INT Major Code: 0X0004, sorted by minor code.
Trace events for INT Major Code: 0X0004 ,sorted by tracepoint.

--------------------------------------------

**Trace Events for INT Major Code: 0X0004, Sorted by Minor Code**
Trace Events for INT Major Code: 0X0004, Sorted by
Tracepoint

(OS) Hardware Interrupt Post-Invocation 00129 (0X0081)
(OS) Hardware Interrupt Post-Invocation 00130 (0X0082)
(OS) Hardware Interrupt Post-Invocation 00131 (0X0083)
(OS) Hardware Interrupt Post-Invocation 00132 (0X0084)
(OS) Hardware Interrupt Post-Invocation 00133 (0X0085)
(OS) Hardware Interrupt Post-Invocation 00134 (0X0086)
(OS) Hardware Interrupt Post-Invocation 00135 (0X0087)
(OS) Hardware Interrupt Post-Invocation 00136 (0X0088)
(OS) Hardware Interrupt Post-Invocation 00137 (0X0089)
(OS) Hardware Interrupt Post-Invocation 00138 (0X008A)
(OS) Hardware Interrupt Post-Invocation 00139 (0X008B)
(OS) Hardware Interrupt Post-Invocation 00140 (0X008C)
(OS) Hardware Interrupt Post-Invocation 00141 (0X008D)
(OS) Hardware Interrupt Post-Invocation 00142 (0X008E)
(OS) Hardware Interrupt Post-Invocation 00143 (0X008F)
(OS) Hardware Interrupt Post-Invocation 00144 (0X0090)
(OS) Hardware Interrupt Post-Invocation 00145 (0X0091)
(OS) Hardware Interrupt Post-Invocation 00146 (0X0092)
(OS) Hardware Interrupt Post-Invocation 00147 (0X0093)
(OS) Hardware Interrupt Post-Invocation 00148 (0X0094)
(OS) Hardware Interrupt Post-Invocation 00149 (0X0095)
(OS) Hardware Interrupt Post-Invocation 00150 (0X0096)
(OS) Hardware Interrupt Post-Invocation 00151 (0X0097)
(OS) Hardware Interrupt Post-Invocation 00152 (0X0098)
(OS) Hardware Interrupt Post-Invocation 00153 (0X0099)
(OS) Hardware Interrupt Post-Invocation 00154 (0X009A)
(OS) Hardware Interrupt Post-Invocation 00155 (0X009B)
(OS) Hardware Interrupt Post-Invocation 00156 (0X009C)
(OS) Hardware Interrupt Post-Invocation 00157 (0X009D)
(OS) Hardware Interrupt Post-Invocation 00158 (0X009E)
(OS) Hardware Interrupt Post-Invocation 00159 (0X009F)
(OS) Hardware Interrupt Post-Invocation 00160 (0X00A0)
(OS) Hardware Interrupt Pre-Invocation 00001 (0X0001)
(OS) Hardware Interrupt Pre-Invocation 00002 (0X0002)
(OS) Hardware Interrupt Pre-Invocation 00003 (0X0003)
(OS) Hardware Interrupt Pre-Invocation 00004 (0X0004)
(OS) Hardware Interrupt Pre-Invocation 00005 (0X0005)
(OS) Hardware Interrupt Pre-Invocation 00006 (0X0006)
(OS) Hardware Interrupt Pre-Invocation 00007 (0X0007)
(OS) Hardware Interrupt Pre-Invocation 00008 (0X0008)
(OS) Hardware Interrupt Pre-Invocation 00009 (0X0009)
(OS) Hardware Interrupt Pre-Invocation 00010 (0X000A)
(OS) Hardware Interrupt Pre-Invocation 00011 (0X000B)
(OS) Hardware Interrupt Pre-Invocation 00012 (0X000C)
(OS) Hardware Interrupt Pre-Invocation 00013 (0X000D)
(OS) Hardware Interrupt Pre-Invocation 00014 (0X000E)
(OS) Hardware Interrupt Pre-Invocation 00015 (0X000F)
(OS) Hardware Interrupt Pre-Invocation 00016 (0X0010)
(OS) Hardware Interrupt Pre-Invocation 00017 (0X0011)
(OS) Hardware Interrupt Pre-Invocation 00018 (0X0012)
(OS) Hardware Interrupt Pre-Invocation 00019 (0X0013)
(OS) Hardware Interrupt Pre-Invocation 00020 (0X0014)
(OS) Hardware Interrupt Pre-Invocation 00021 (0X0015)
(OS) Hardware Interrupt Pre-Invocation 00022 (0X0016)
(OS) Hardware Interrupt Pre-Invocation 00023 (0X0017)
(OS) Hardware Interrupt Pre-Invocation 00024 (0X0018)
(OS) Hardware Interrupt Pre-Invocation 00025 (0X0019)
(OS) Hardware Interrupt Pre-Invocation 00026 (0X001A)
(OS) Hardware Interrupt Pre-Invocation 00027 (0X001B)
(OS) Hardware Interrupt Pre-Invocation 00028 (0X001C)
(OS) Hardware Interrupt Pre-Invocation 00029 (0X001D)
(OS) Hardware Interrupt Pre-Invocation 00030 (0X001E)
(OS) Hardware Interrupt Pre-Invocation 00031 (0X001F)
(OS) Hardware Interrupt Pre-Invocation 00032 (0X0020)
INT Major Code: 0X0004 Minor Code: 1 (0X0001)

Description  
(OS) Hardware Interrupt Pre-Invocation

Tracepoint  
Static tracepoint in INT.

Minor Code  
1 (0X0001)

Trace Groups  
No groups assigned.

Trace Types  
PRE

Traced Parameters  
Interrupt Level=0 (Timer)

INT Major Code: 0X0004 Minor Code: 2 (0X0002)

Description  
(OS) Hardware Interrupt Pre-Invocation

Tracepoint  
Static tracepoint in INT.

Minor Code  
2 (0X0002)

Trace Groups  
No groups assigned.

Trace Types  
PRE

Traced Parameters  
Interrupt Level=1 (Keyboard)

INT Major Code: 0X0004 Minor Code: 3 (0X0003)

Description  
(OS) Hardware Interrupt Pre-Invocation

Tracepoint  
Static tracepoint in INT.
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
  Interrupt Level=2 (NMI)

--------------------------------------------
INT Major Code: 0X0004 Minor Code: 4 (0X0004)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 4 (0X0004)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
  Interrupt Level=3 (Serial Port 2)

--------------------------------------------
INT Major Code: 0X0004 Minor Code: 5 (0X0005)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 5 (0X0005)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
  Interrupt Level=4 (Serial Port 1)
INT Major Code: 0X0004 Minor Code: 6 (0X0006)

**Description**  
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**  
Static tracepoint in INT.

**Minor Code**  
6 (0X0006)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
Interrupt Level=5 (Parallel Port 2)

INT Major Code: 0X0004 Minor Code: 7 (0X0007)

**Description**  
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**  
Static tracepoint in INT.

**Minor Code**  
7 (0X0007)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
Interrupt Level=6 (Diskette Controller)

INT Major Code: 0X0004 Minor Code: 8 (0X0008)

**Description**  
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**  
Static tracepoint in INT.
Minor Code 8 (0X0008)

Trace Groups No groups assigned.

Trace Types PRE

Traced Parameters

Interrupt Level=7 (Parallel Port 1)

INT Major Code: 0X0004 Minor Code: 9 (0X0009)

Description (OS) Hardware Interrupt Pre-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 9 (0X0009)

Trace Groups No groups assigned.

Trace Types PRE

Traced Parameters

Interrupt Level=8 (Realtime Clock)

INT Major Code: 0X0004 Minor Code: 10 (0X000A)

Description (OS) Hardware Interrupt Pre-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 10 (0X000A)

Trace Groups No groups assigned.

Trace Types PRE

Traced Parameters

Interrupt Level=9
INT Major Code: 0X0004 Minor Code: 11 (0X000B)

**Description**
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
11 (0X000B)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Interrupt Level=A

INT Major Code: 0X0004 Minor Code: 12 (0X000C)

**Description**
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
12 (0X000C)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Interrupt Level=B

INT Major Code: 0X0004 Minor Code: 13 (0X000D)

**Description**
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**
Static tracepoint in INT.
Minor Code: 13 (0X000D)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:

Interrupt Level=C

INT Major Code: 0X0004 Minor Code: 14 (0X000E)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 14 (0X000E)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:

Interrupt Level=D (Coprocessor)

INT Major Code: 0X0004 Minor Code: 15 (0X000F)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 15 (0X000F)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:

Interrupt Level=E (Fixed Disk Controller)
### INT Major Code: 0X0004 Minor Code: 16 (0X0010)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Hardware Interrupt Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in INT.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>16 (0X0010)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Interrupt Level=F</td>
</tr>
</tbody>
</table>

### INT Major Code: 0X0004 Minor Code: 17 (0X0011)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Hardware Interrupt Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in INT.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>17 (0X0011)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Interrupt Level=10</td>
</tr>
</tbody>
</table>

### INT Major Code: 0X0004 Minor Code: 18 (0X0012)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Hardware Interrupt Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in INT.</td>
</tr>
</tbody>
</table>
Minor Code: 18 (0X0012)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters: Interrupt Level=11

INT Major Code: 0X0004 Minor Code: 19 (0X0013)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 19 (0X0013)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters: Interrupt Level=12

INT Major Code: 0X0004 Minor Code: 20 (0X0014)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 20 (0X0014)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters: Interrupt Level=13
<table>
<thead>
<tr>
<th>INT Major Code: 0X0004 Minor Code: 21 (0X0015)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT Major Code: 0X0004 Minor Code: 22 (0X0016)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT Major Code: 0X0004 Minor Code: 23 (0X0017)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
</tbody>
</table>
Minor Code: 23 (0X0017)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
   Interrupt Level=16

INT Major Code: 0X0004
Minor Code: 24 (0X0018)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 24 (0X0018)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
   Interrupt Level=17

INT Major Code: 0X0004
Minor Code: 25 (0X0019)

Description: (OS) Hardware Interrupt Pre-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 25 (0X0019)
Trace Groups: No groups assigned.
Trace Types: PRE
Traced Parameters:
   Interrupt Level=18
<table>
<thead>
<tr>
<th>Major Code</th>
<th>Minor Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X0004</td>
<td>0X001A</td>
<td>(OS) Hardware Interrupt Pre-Invocation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracepoint: Static tracepoint in INT.</td>
</tr>
<tr>
<td>Minor Code</td>
<td></td>
<td>26 (0X001A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td></td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
<td>Interrupt Level=19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Code</th>
<th>Minor Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X0004</td>
<td>0X001B</td>
<td>(OS) Hardware Interrupt Pre-Invocation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracepoint: Static tracepoint in INT.</td>
</tr>
<tr>
<td>Minor Code</td>
<td></td>
<td>27 (0X001B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td></td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
<td>Interrupt Level=1A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Code</th>
<th>Minor Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X0004</td>
<td>0X001C</td>
<td>(OS) Hardware Interrupt Pre-Invocation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracepoint: Static tracepoint in INT.</td>
</tr>
</tbody>
</table>
Minor Code: 28 (0X001C)

Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters: Interrupt Level=1B

INT Major Code: 0X0004 Minor Code: 29 (0X001D)

Description: (OS) Hardware Interrupt Pre-Invocation

Tracepoint: Static tracepoint in INT.

Minor Code: 29 (0X001D)

Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters: Interrupt Level=1C

INT Major Code: 0X0004 Minor Code: 30 (0X001E)

Description: (OS) Hardware Interrupt Pre-Invocation

Tracepoint: Static tracepoint in INT.

Minor Code: 30 (0X001E)

Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters: Interrupt Level=1D
INT Major Code: 0X0004 Minor Code: 31 (0X001F)

**Description**  
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**  
Static tracepoint in INT.

**Minor Code**  
31 (0X001F)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**

Interrupt Level=1E

INT Major Code: 0X0004 Minor Code: 32 (0X0020)

**Description**  
(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**  
Static tracepoint in INT.

**Minor Code**  
32 (0X0020)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**

Interrupt Level=1F

INT Major Code: 0X0004 Minor Code: 129 (0X0081)

**Description**  
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**  
Static tracepoint in INT.
INT Major Code: 0X0004 Minor Code: 130 (0X0082)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 130 (0X0082)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
  Interrupt Level=1 (Keyboard)

INT Major Code: 0X0004 Minor Code: 131 (0X0083)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 131 (0X0083)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
  Interrupt Level=2 (NMI)
INT Major Code: 0X0004 Minor Code: 132 (0X0084)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 132 (0X0084)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
  Interrupt Level=3 (Serial Port 2)

INT Major Code: 0X0004 Minor Code: 133 (0X0085)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 133 (0X0085)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
  Interrupt Level=4 (Serial Port 1)

INT Major Code: 0X0004 Minor Code: 134 (0X0086)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
INT Major Code: 0X0004 Minor Code: 134 (0X0086)

Description
(OS) Hardware Interrupt Post-Invocation

Tracepoint
Static tracepoint in INT.

INT Major Code: 0X0004 Minor Code: 135 (0X0087)

Description
(OS) Hardware Interrupt Post-Invocation

Tracepoint
Static tracepoint in INT.

INT Major Code: 0X0004 Minor Code: 136 (0X0088)

Description
(OS) Hardware Interrupt Post-Invocation

Tracepoint
Static tracepoint in INT.
INT Major Code: 0X0004 Minor Code: 137 (0X0089)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
137 (0X0089)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Interrupt Level=8 (Realtime Clock)

INT Major Code: 0X0004 Minor Code: 138 (0X008A)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
138 (0X008A)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Interrupt Level=9

INT Major Code: 0X0004 Minor Code: 139 (0X008B)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.
Minor Code: 139 (0X008B)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
 Interrupt Level=A

INT Major Code: 0X0004 Minor Code: 140 (0X008C)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 140 (0X008C)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
 Interrupt Level=B

INT Major Code: 0X0004 Minor Code: 141 (0X008D)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 141 (0X008D)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters:
 Interrupt Level=C
<table>
<thead>
<tr>
<th>Major Code</th>
<th>Minor Code</th>
<th>Description</th>
<th>Tracepoint</th>
<th>Minor Code</th>
<th>Trace Groups</th>
<th>Trace Types</th>
<th>Traced Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X0004</td>
<td>142 (0X008E)</td>
<td>(OS) Hardware Interrupt Post-Invocation</td>
<td>Static tracepoint in INT.</td>
<td>142 (0X008E)</td>
<td>No groups assigned.</td>
<td>POST</td>
<td>Interrupt Level=D (Coprocessor)</td>
</tr>
<tr>
<td>0X0004</td>
<td>143 (0X008F)</td>
<td>(OS) Hardware Interrupt Post-Invocation</td>
<td>Static tracepoint in INT.</td>
<td>143 (0X008F)</td>
<td>No groups assigned.</td>
<td>POST</td>
<td>Interrupt Level=E (Fixed Disk Controller)</td>
</tr>
<tr>
<td>0X0004</td>
<td>144 (0X0090)</td>
<td>(OS) Hardware Interrupt Post-Invocation</td>
<td>Static tracepoint in INT.</td>
<td></td>
<td></td>
<td>POST</td>
<td></td>
</tr>
</tbody>
</table>
Minor Code: 144 (0X0090)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters: Interrupt Level=F

INT Major Code: 0X0004 Minor Code: 145 (0X0091)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 145 (0X0091)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters: Interrupt Level=10

INT Major Code: 0X0004 Minor Code: 146 (0X0092)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 146 (0X0092)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters: Interrupt Level=11
## INT Major Code: 0X0004 Minor Code: 147 (0X0093)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
147 (0X0093)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Interrupt Level=12

## INT Major Code: 0X0004 Minor Code: 148 (0X0094)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.

**Minor Code**
148 (0X0094)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Interrupt Level=13

## INT Major Code: 0X0004 Minor Code: 149 (0X0095)

**Description**
(OS) Hardware Interrupt Post-Invocation

**Tracepoint**
Static tracepoint in INT.
Minor Code 149 (0X0095)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters

Interrupt Level=14

INT Major Code: 0X0004 Minor Code: 150 (0X0096)

Description (OS) Hardware Interrupt Post-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 150 (0X0096)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters

Interrupt Level=15

INT Major Code: 0X0004 Minor Code: 151 (0X0097)

Description (OS) Hardware Interrupt Post-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 151 (0X0097)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters

Interrupt Level=16
INT Major Code: 0X0004 Minor Code: 152 (0X0098)

Description  (OS) Hardware Interrupt Post-Invocation
Tracepoint  Static tracepoint in INT.
Minor Code  152 (0X0098)
Trace Groups  No groups assigned.
Trace Types  POST
Traced Parameters

Interrupt Level=17

INT Major Code: 0X0004 Minor Code: 153 (0X0099)

Description  (OS) Hardware Interrupt Post-Invocation
Tracepoint  Static tracepoint in INT.
Minor Code  153 (0X0099)
Trace Groups  No groups assigned.
Trace Types  POST
Traced Parameters

Interrupt Level=18

INT Major Code: 0X0004 Minor Code: 154 (0X009A)

Description  (OS) Hardware Interrupt Post-Invocation
Tracepoint  Static tracepoint in INT.
Minor Code 154 (0X009A)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters Interrupt Level=19

--------------------------------------------

INT Major Code: 0X0004 Minor Code: 155 (0X009B)

Description (OS) Hardware Interrupt Post-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 155 (0X009B)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters Interrupt Level=1A

--------------------------------------------

INT Major Code: 0X0004 Minor Code: 156 (0X009C)

Description (OS) Hardware Interrupt Post-Invocation

Tracepoint Static tracepoint in INT.

Minor Code 156 (0X009C)

Trace Groups No groups assigned.

Trace Types POST

Traced Parameters Interrupt Level=1B
INT Major Code: 0X0004 Minor Code: 157 (0X009D)

**Description**  (OS) Hardware Interrupt Post-Invocation

**Tracepoint**  Static tracepoint in INT.

**Minor Code**  157 (0X009D)

**Trace Groups**  No groups assigned.

**Trace Types**  POST

**Traced Parameters**

Interrupt Level=1C

INT Major Code: 0X0004 Minor Code: 158 (0X009E)

**Description**  (OS) Hardware Interrupt Post-Invocation

**Tracepoint**  Static tracepoint in INT.

**Minor Code**  158 (0X009E)

**Trace Groups**  No groups assigned.

**Trace Types**  POST

**Traced Parameters**

Interrupt Level=1D

INT Major Code: 0X0004 Minor Code: 159 (0X009F)

**Description**  (OS) Hardware Interrupt Post-Invocation

**Tracepoint**  Static tracepoint in INT.
Minor Code: 159 (0X009F)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters: Interrupt Level=1E

---

INT Major Code: 0X0004 Minor Code: 160 (0X00A0)

Description: (OS) Hardware Interrupt Post-Invocation
Tracepoint: Static tracepoint in INT.
Minor Code: 160 (0X00A0)
Trace Groups: No groups assigned.
Trace Types: POST
Traced Parameters: Interrupt Level=1F

---

KERNEL Services Trace Events

The external API tracepoints for the KERNEL major code are identified in the following tables. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace Events for OS2KRNL Major Code: 0X0005, sorted by minor code.
Trace Events for OS2KRNL Major Code: 0X0005, sorted by tracepoint.
Kernel API Tracepoints Indirected Via DOSCALL1.

---

Trace Events for OS2KRNL Major Code: 0X0005, Sorted by Minor Code
00001 (0X0001) (OS) DosEnterCritSec Post-Invocation
00002 (0X0002) (OS) DosExitCritSec Post-Invocation
00003 (0X0003) (OS) DosHoldSignal Post-Invocation (2)
00006 (0X0006) (OS) Dos32AliasMem Post-Invocation
00007 (0X0007) (OS) Dos32AllocMem Post-Invocation
00008 (0X0008) (OS) Dos32AllocProtectedMem Post-Invocation
00009 (0X0009) (OS) Dos32AllocSharedMem Post-Invocation
00010 (0X000A) (OS) Dos32CreateThread Post-Invocation
00011 (0X000B) (OS) Dos32Debug Dos Post-Invocation
00012 (0X000C) (OS) Dos32ExitList Dos Post-Invocation
00013 (0X000D) (OS) Dos32FreeMem Post-Invocation
00014 (0X000E) (OS) Dos32GetNamedSharedMem Post-Invocation
00015 (0X000F) (OS) Dos32GetSharedMem Post-Invocation
00016 (0X0010) (OS) Dos32GiveSharedMem Post-Invocation
00017 (0X0011) (OS) Dos32QueryMem Post-Invocation
00018 (0X0012) (OS) Dos32QueryMemState Post-Invocation
00019 (0X0013) (OS) Dos32SetMem Post-Invocation
00020 (0X0014) (OS) DosAllocHuge Post-Invocation
00021 (0X0015) (OS) DosAllocProtHuge Post-Invocation
00022 (0X0016) (OS) DosAllocProtSeg Post-Invocation
00023 (0X0017) (OS) DosAllocSeg Post-Invocation
00024 (0X0018) (OS) DosAllocShrProtSeg Post-Invocation
00025 (0X0019) (OS) DosAllocShrSeg Post-Invocation
00026 (0X001A) (OS) DosBeep Post-Invocation
00027 (0X001B) (OS) DosBufReset Post-Invocation
00028 (0X001C) (OS) DosCallNmPipe Post-Invocation
00029 (0X001D) (OS) DosChDir Post-Invocation
00030 (0X001E) (OS) DosChgFilePtr Post-Invocation
00031 (0X001F) (OS) DosClrAccess Post-Invocation
00032 (0X0020) (OS) DosClose Post-Invocation
00033 (0X0021) (OS) DosCloseSem Post-Invocation
00034 (0X0022) (OS) DosConnectNmPipe Post-Invocation
00035 (0X0023) (OS) DosCreateCSAlias Post-Invocation
00036 (0X0024) (OS) DosCreateSem Post-Invocation
00037 (0X0025) (OS) DosCreateThread Post-Invocation
00038 (0X0026) (OS) DosCWait Post-Invocation
00039 (0X0027) (OS) DosDeletePost-Invocation
00040 (0X0028) (OS) DosDevConfig Post-Invocation
00041 (0X0029) (OS) DosDevIoctl2 Post-Invocation
00042 (0X002A) (OS) DosDevIoctl Post-Invocation
00043 (0X002B) (OS) DosDisConnectNmPipe Post-Invocation
00044 (0X002C) (OS) DosDupHandle Post-Invocation
00045 (0X002D) (OS) DosEditName Post-Invocation
00047 (0X002F) (OS) DosError Pre-Invocation
00048 (0X0031) (OS) Dos32ExitDos Post-Invocation
00051 (0X0033) (OS) DosExitDosList Dos Post-Invocation
00052 (0X0034) (OS) DosFileIo Post-Invocation
00053 (0X0035) (OS) DosFileLocks Post-Invocation
00054 (0X0036) (OS) DosFindClose Post-Invocation
00055 (0X0037) (OS) DosFindFirst2 Post-Invocation
00056 (0X0038) (OS) DosFindFirst Post-Invocation
00057 (0X0039) (OS) DosFindFromName Post-Invocation
00058 (0X003A) (OS) DosFindNext Post-Invocation
00059 (0X003B) (OS) DosFindNotifyClose Post-Invocation
00060 (0X003C) (OS) DosFindNotifyFirst Post-Invocation
00061 (0X003D) (OS) DosFindNotifyNext Post-Invocation
00062 (0X003E) (OS) DosFlagProcess Post-Invocation
00063 (0X003F) (OS) DosFreeModule Post-Invocation
00064 (0X0040) (OS) DosFreeSeg Post-Invocation
00065 (0X0041) (OS) DosFSAttach Post-Invocation
00066 (0X0042) (OS) DosFSCtl Post-Invocation
00067 (0X0043) (OS) DosGetDateTime Post-Invocation
00068 (0X0044) (OS) DosGetModHandle Post-Invocation
00069 (0X0045) (OS) DosGetModuleName Post-Invocation
00070 (0X0046) (OS) DosGetPid Post-Invocation
00071 (0X0047) (OS) DosGetProcAddr Post-Invocation
00072 (0X0048) (OS) DosGetPrty Post-Invocation
00073 (0X0049) (OS) DosGetResource Post-Invocation
00074 (0X004A) (OS) DosGetSeg Post-Invocation
00075 (0X004B) (OS) DosGetShrSeg Post-Invocation
00076 (0X004C) (OS) DosGetVersion Post-Invocation
00077 (0X004D) (OS) DosGiveSeg Post-Invocation
00078 (0X004E) (OS) DosHoldSignal Post-Invocation
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00079</td>
<td>(0X004F) (OS) DosICopy Post-Invocation</td>
</tr>
<tr>
<td>00080</td>
<td>(0X0050) (OS) DosExecPgm Post-Invocation</td>
</tr>
<tr>
<td>00083</td>
<td>(0X0053) (OS) DosSetCP Post-Invocation</td>
</tr>
<tr>
<td>00084</td>
<td>(0X0054) (OS) DosKillProcess Post-Invocation</td>
</tr>
<tr>
<td>00085</td>
<td>(0X0055) (OS) DosLoadModule Post-Invocation</td>
</tr>
<tr>
<td>00086</td>
<td>(0X0056) (OS) DosMakeNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00087</td>
<td>(0X0057) (OS) DosMakePipe Post-Invocation</td>
</tr>
<tr>
<td>00088</td>
<td>(0X0058) (OS) DosMkDir2 Post-Invocation</td>
</tr>
<tr>
<td>00089</td>
<td>(0X0059) (OS) DosMkDir Post-Invocation</td>
</tr>
<tr>
<td>00095</td>
<td>(0X005F) (OS) DosMove Post-Invocation</td>
</tr>
<tr>
<td>00096</td>
<td>(0X0060) (OS) DosMuxSemWait Post-Invocation (Waited)</td>
</tr>
<tr>
<td>00097</td>
<td>(0X0061) (OS) DosNewSize Post-Invocation</td>
</tr>
<tr>
<td>00098</td>
<td>(0X0062) (OS) DosOpen2 Post-Invocation</td>
</tr>
<tr>
<td>00099</td>
<td>(0X0063) (OS) DosOpen Post-Invocation</td>
</tr>
<tr>
<td>00100</td>
<td>(0X0064) (OS) DosOpenSem Post-Invocation</td>
</tr>
<tr>
<td>00101</td>
<td>(0X0065) (OS) DosOpLockRelease Post-Invocation</td>
</tr>
<tr>
<td>00102</td>
<td>(0X0066) (OS) DosOpLockWait Post-Invocation</td>
</tr>
<tr>
<td>00103</td>
<td>(0X0067) (OS) DosPeekNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00104</td>
<td>(0X0068) (OS) DosPhysicalDisk Post-Invocation</td>
</tr>
<tr>
<td>00105</td>
<td>(0X0069) (OS) DosQNmPipeState Post-Invocation</td>
</tr>
<tr>
<td>00106</td>
<td>(0X006A) (OS) DosPortAccess Post-Invocation</td>
</tr>
<tr>
<td>00108</td>
<td>(0X006C) (OS) DosQCurDir Post-Invocation</td>
</tr>
<tr>
<td>00109</td>
<td>(0X006D) (OS) DosQCurDisk Post-Invocation</td>
</tr>
<tr>
<td>00110</td>
<td>(0X006E) (OS) DosQFHandState Post-Invocation</td>
</tr>
<tr>
<td>00111</td>
<td>(0X006F) (OS) DosQFileInfo Post-Invocation</td>
</tr>
<tr>
<td>00112</td>
<td>(0X0070) (OS) DosQFileMode Post-Invocation</td>
</tr>
<tr>
<td>00113</td>
<td>(0X0071) (OS) DosQFSAttach Post-Invocation</td>
</tr>
<tr>
<td>00114</td>
<td>(0X0072) (OS) DosQFSInfo Post-Invocation</td>
</tr>
<tr>
<td>00115</td>
<td>(0X0073) (OS) DosQHandType Post-Invocation</td>
</tr>
<tr>
<td>00116</td>
<td>(0X0074) (OS) DosQNmPHandState Post-Invocation</td>
</tr>
<tr>
<td>00117</td>
<td>(0X0075) (OS) DosQNmPipeInfo Post-Invocation</td>
</tr>
<tr>
<td>00118</td>
<td>(0X0076) (OS) DosQPathInfo Post-Invocation</td>
</tr>
<tr>
<td>00119</td>
<td>(0X0077) (OS) DosQSysInfo Post-Invocation</td>
</tr>
<tr>
<td>00120</td>
<td>(0X0078) (OS) DosQVerify Post-Invocation</td>
</tr>
<tr>
<td>00121</td>
<td>(0X0079) (OS) DosRawReadNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00122</td>
<td>(0X007A) (OS) DosRawWriteNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00123</td>
<td>(0X007B) (OS) DosRead Post-Invocation</td>
</tr>
<tr>
<td>00125</td>
<td>(0X007D) (OS) DosReallocHuge Post-Invocation</td>
</tr>
<tr>
<td>00126</td>
<td>(0X007E) (OS) DosReallocSeg Post-Invocation</td>
</tr>
<tr>
<td>00127</td>
<td>(0X007F) (OS) Dos32ResumeThread Dos Post-Invocation</td>
</tr>
<tr>
<td>00128</td>
<td>(0X0080) (OS) DosRmDir Post-Invocation</td>
</tr>
<tr>
<td>00129</td>
<td>(0X0081) (OS) DosSelectDisk Post-Invocation</td>
</tr>
<tr>
<td>00132</td>
<td>(0X0085) (OS) DosSemWait Post-Invocation</td>
</tr>
<tr>
<td>00135</td>
<td>(0X0087) (OS) DosSendSignal Post-Invocation</td>
</tr>
<tr>
<td>00137</td>
<td>(0X0089) (OS) DosSetDate time Post-Invocation</td>
</tr>
<tr>
<td>00138</td>
<td>(0X008A) (OS) DosSetFHandState Post-Invocation</td>
</tr>
<tr>
<td>00139</td>
<td>(0X008B) (OS) DosSetFileInfo Post-Invocation</td>
</tr>
<tr>
<td>00140</td>
<td>(0X008C) (OS) DosSetFileMode Post-Invocation</td>
</tr>
<tr>
<td>00141</td>
<td>(0X008D) (OS) DosSetFSInfo Post-Invocation</td>
</tr>
<tr>
<td>00142</td>
<td>(0X008E) (OS) DosSetMaxFH Post-Invocation</td>
</tr>
<tr>
<td>00143</td>
<td>(0X008F) (OS) DosSetNmPHandState Post-Invocation</td>
</tr>
<tr>
<td>00144</td>
<td>(0X0090) (OS) DosSetNmPipeSem Post-Invocation</td>
</tr>
<tr>
<td>00145</td>
<td>(0X0091) (OS) DosSetPathInfo Post-Invocation</td>
</tr>
<tr>
<td>00146</td>
<td>(0X0092) (OS) DosSetPrty Post-Invocation</td>
</tr>
<tr>
<td>00147</td>
<td>(0X0093) (OS) DosSetSigHandler Post-Invocation</td>
</tr>
<tr>
<td>00148</td>
<td>(0X0094) (OS) DosSetVec Post-Invocation</td>
</tr>
<tr>
<td>00149</td>
<td>(0X0095) (OS) DosSetVerify Post-Invocation</td>
</tr>
<tr>
<td>00150</td>
<td>(0X0096) (OS) DosSleep Post-Invocation</td>
</tr>
<tr>
<td>00151</td>
<td>(0X0097) (OS) Dos32SuspendThread Dos Post-Invocation</td>
</tr>
<tr>
<td>00152</td>
<td>(0X0098) (OS) DosSystemService Post-Invocation</td>
</tr>
<tr>
<td>00153</td>
<td>(0X0099) (OS) DosTimerAsync Post-Invocation</td>
</tr>
<tr>
<td>00154</td>
<td>(0X009A) (OS) DosTimerStart Post-Invocation</td>
</tr>
<tr>
<td>00155</td>
<td>(0X009B) (OS) Dos32StopTimer Post-Invocation</td>
</tr>
<tr>
<td>00156</td>
<td>(0X009C) (OS) DosTransactNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00157</td>
<td>(0X009D) (OS) DosWaitNmPipe Post-Invocation</td>
</tr>
<tr>
<td>00158</td>
<td>(0X009E) (OS) DosWrite Post-Invocation</td>
</tr>
<tr>
<td>00163</td>
<td>(0X00A3) (OS) Dos32AliasMem Pre-Invocation</td>
</tr>
<tr>
<td>00164</td>
<td>(0X00A4) (OS) Dos32AllocMem Pre-Invocation</td>
</tr>
<tr>
<td>00165</td>
<td>(0X00A5) (OS) Dos32AllocProtectedMem Pre-Invocation</td>
</tr>
<tr>
<td>00166</td>
<td>(0X00A6) (OS) Dos32AllocSharedMem Pre-Invocation</td>
</tr>
<tr>
<td>00167</td>
<td>(0X00A7) (OS) Dos32CreateThread Pre-Invocation</td>
</tr>
<tr>
<td>00168</td>
<td>(0X00A8) (OS) Dos32Debug Dos Pre-Invocation</td>
</tr>
<tr>
<td>00169</td>
<td>(0X00A9) (OS) Dos32ExitList Dos Pre-Invocation</td>
</tr>
</tbody>
</table>
00170 (0X00AA) (OS) Dos32FreeMem Pre-Invocation
00171 (0X00AB) (OS) Dos32GetNamedSharedMem Pre-Invocation
00172 (0X00AC) (OS) Dos32GetSharedMem Pre-Invocation
00173 (0X00AD) (OS) Dos32GiveSharedMem Pre-Invocation
00174 (0X00AE) (OS) Dos32QueryMem Pre-Invocation
00175 (0X00AF) (OS) Dos32QueryMemState Pre-Invocation
00176 (0X00B0) (OS) Dos32SetMem Pre-Invocation
00177 (0X00B1) (OS) DosAllocHuge Pre-Invocation
00178 (0X00B2) (OS) DosAllocProtHuge Pre-Invocation
00179 (0X00B3) (OS) DosAllocProtSeg Pre-Invocation
00180 (0X00B4) (OS) DosAllocSeg Pre-Invocation
00181 (0X00B5) (OS) DosAllocShrProtSeg Pre-Invocation
00182 (0X00B6) (OS) DosAllocShrSeg Pre-Invocation
00183 (0X00B7) (OS) DosBeep Pre-Invocation
00184 (0X00B8) (OS) DosBufReset Pre-Invocation
00185 (0X00B9) (OS) DosCallNmPipe Pre-Invocation
00186 (0X00BA) (OS) DosChDir Pre-Invocation
00187 (0X00BB) (OS) DosChgFilePtr Pre-Invocation
00188 (0X00BC) (OS) DosCliAccess Pre-Invocation
00189 (0X00BD) (OS) DosClose Pre-Invocation
00190 (0X00BE) (OS) DosCloseSem Pre-Invocation
00191 (0X00BF) (OS) DosConnectNmPipe Pre-Invocation
00192 (0X00C0) (OS) DosCreateCSAlias Pre-Invocation
00193 (0X00C1) (OS) DosCreateSem Pre-Invocation
00194 (0X00C2) (OS) DosCreateThread Pre-Invocation
00195 (0X00C3) (OS) DosCWait Pre-Invocation
00196 (0X00C4) (OS) DosDelete Pre-Invocation
00197 (0X00C5) (OS) DosDevConfig Pre-Invocation
00198 (0X00C6) (OS) DosDevIoctl2 Pre-Invocation
00199 (0X00C7) (OS) DosDevIoctl Pre-Invocation
00200 (0X00C8) (OS) DosDisConnectNmPipe Pre-Invocation
00201 (0X00C9) (OS) DosDupHandle Pre-Invocation
00202 (0X00CA) (OS) DosEditName Pre-Invocation
00203 (0X00CB) (OS) DosEnterCritSec Pre-Invocation
00204 (0X00CC) (OS) DosError Pre-Invocation
00206 (0X00CE) (OS) DosExit Dos Pre-Invocation
00207 (0X00CF) (OS) DosExitCritSec Pre-Invocation
00208 (0X00D0) (OS) DosExitList Dos Pre-Invocation
00209 (0X00D1) (OS) DosFileIO Pre-Invocation
00210 (0X00D2) (OS) DosFileLocks Pre-Invocation
00211 (0X00D3) (OS) DosFindClose Pre-Invocation
00212 (0X00D4) (OS) DosFindFirst2 Pre-Invocation
00213 (0X00D5) (OS) DosFindFirst Pre-Invocation
00214 (0X00D6) (OS) DosFindFromName Pre-Invocation
00215 (0X00D7) (OS) DosFindNext Pre-Invocation
00216 (0X00D8) (OS) DosFindNotifyClose Pre-Invocation
00217 (0X00D9) (OS) DosFindNotifyFirst Pre-Invocation
00218 (0X00DA) (OS) DosFindNotifyNext Pre-Invocation
00219 (0X00DB) (OS) DosFlagProcess Pre-Invocation
00220 (0X00DC) (OS) DosFreeModule Pre-Invocation
00221 (0X00DD) (OS) DosFreeSeg Pre-Invocation
00222 (0X00DE) (OS) DosFSAttach Pre-Invocation
00223 (0X00DF) (OS) DosFSCTL Pre-Invocation
00224 (0X00E0) (OS) DosGetDateTime Pre-Invocation
00225 (0X00E1) (OS) DosGetModHandle Pre-Invocation
00226 (0X00E2) (OS) DosGetModuleName Pre-Invocation
00227 (0X00E3) (OS) DosGetPid Pre-Invocation
00228 (0X00E4) (OS) DosGetProcAddr Pre-Invocation
00229 (0X00E5) (OS) DosGetPrty Pre-Invocation
00230 (0X00E6) (OS) DosGetResource Pre-Invocation
00231 (0X00E7) (OS) DosGetSeg Pre-Invocation
00232 (0X00E8) (OS) DosGetShrSeg Pre-Invocation
00233 (0X00E9) (OS) DosGetVersion Pre-Invocation
00234 (0X00EA) (OS) DosGiveSeg Pre-Invocation
00235 (0X00EB) (OS) DosHoldSignal Pre-Invocation
00236 (0X00EC) (OS) DosCopy Pre-Invocation
00237 (0X00ED) (OS) DosExecPgm Pre-Invocation
00240 (0X00F0) (OS) DosSetCP Pre-Invocation
00241 (0X00F1) (OS) DosKillProcess Pre-Invocation
00242 (0X00F2) (OS) DosLoadModule Pre-Invocation
00243 (0X00F3) (OS) DosMakeNmPipe Pre-Invocation
00244 (0X00F4) (OS) DosMakePipe Pre-Invocation
00245 (0X00F5) (OS) DosMkDir2 Pre-Invocation
00340 (0X0154) (OS) Dos32CloseEventSem Post-Invocation
00341 (0X0155) (OS) Dos32ResetEventSem Pre-Invocation
00342 (0X0156) (OS) Dos32ResetEventSem Post-Invocation
00343 (0X0157) (OS) Dos32PostEventSem Pre-Invocation
00344 (0X0158) (OS) Dos32PostEventSem Post-Invocation
00345 (0X0159) (OS) Dos32QueryEventSem Pre-Invocation
00346 (0X015A) (OS) Dos32QueryEventSem Post-Invocation
00347 (0X015B) (OS) Dos32CreateMutexSem Pre-Invocation
00348 (0X015C) (OS) Dos32CreateMutexSem Post-Invocation
00349 (0X015D) (OS) Dos32OpenMutexSem Pre-Invocation
00350 (0X015E) (OS) Dos32OpenMutexSem Post-Invocation
00351 (0X015F) (OS) Dos32CloseMutexSem Pre-Invocation
00352 (0X0160) (OS) Dos32CloseMutexSem Post-Invocation
00353 (0X0161) (OS) Dos32RequestMutexSem Pre-Invocation
00354 (0X0162) (OS) Dos32RequestMutexSem Post-Invocation
00355 (0X0163) (OS) Dos32ReleaseMutexSem Pre-Invocation
00356 (0X0164) (OS) Dos32ReleaseMutexSem Post-Invocation
00357 (0X0165) (OS) Dos32CreateMuxWaitSem Pre-Invocation
00358 (0X0166) (OS) Dos32CreateMuxWaitSem Post-Invocation
00359 (0X0167) (OS) Dos32OpenMuxWaitSem Pre-Invocation
00360 (0X0168) (OS) Dos32OpenMuxWaitSem Post-Invocation
00361 (0X0169) (OS) Dos32CloseMuxWaitSem Pre-Invocation
00362 (0X016A) (OS) Dos32CloseMuxWaitSem Post-Invocation
00363 (0X016B) (OS) Dos32CreateMuxWaitSem Post-Invocation
00364 (0X016C) (OS) Dos32OpenMuxWaitSem Post-Invocation
00365 (0X016D) (OS) Dos32CloseMuxWaitSem Pre-Invocation
00366 (0X016E) (OS) Dos32CloseMuxWaitSem Post-Invocation
00367 (0X016F) (OS) Dos32WaitMuxWaitSem Pre-Invocation
00368 (0X0170) (OS) Dos32WaitMuxWaitSem Post-Invocation
00369 (0X0171) (OS) Dos32AddMuxWaitSem Pre-Invocation
00370 (0X0172) (OS) Dos32AddMuxWaitSem Post-Invocation
00371 (0X0173) (OS) Dos32DeleteMuxWaitSem Pre-Invocation
00372 (0X0174) (OS) Dos32DeleteMuxWaitSem Post-Invocation
00373 (0X0175) (OS) Dos32QueryMuxWaitSem Pre-Invocation
00374 (0X0176) (OS) Dos32QueryMuxWaitSem Post-Invocation
00375 (0X0177) (OS) DosGetCP Pre-Invocation
00376 (0X0178) (OS) DosGetCP Post-Invocation
00377 (0X0179) (OS) Dos32AsyncTimer Pre-Invocation
00378 (0X017A) (OS) Dos32AsyncTimer Post-Invocation
00379 (0X017B) (OS) Dos32StartTimer Pre-Invocation
00380 (0X017C) (OS) Dos32StartTimer Post-Invocation
00391 (0X0187) (OS) Dos32WaitThread Pre-Invocation
00392 (0X0188) (OS) Dos32WaitThread Post-Invocation
00393 (0X0189) (OS) Cluster Allocate Pre-Invocation
00394 (0X018A) (OS) Cluster Allocate Post-Invocation
00395 (0X018B) (OS) Cluster Release Pre-Invocation
00396 (0X018C) (OS) Cluster Release Post-Invocation
00401 (0X0191) (OS) File Lock/Unlock Pre-Invocation
00402 (0X0192) (OS) File Lock/Unlock Post-Invocation
00403 (0X0193) (OS) DosMuxSemWait Post-Invocation (No Wait)
00404 (0X0194) (OS) DosEnumAttribute Pre-Invocation
00405 (0X0195) (OS) DosEnumAttribute Post-Invocation
00406 (0X0196) (OS) DosSetFileInfo Pre-Invocation
00407 (0X0197) (OS) DosSetFileInfo Post-Invocation
00408 (0X0198) (OS) DosSetPathInfo Pre-Invocation
00409 (0X0199) (OS) DosSetPathInfo Post-Invocation
00410 (0X019A) (OS) Dos32QueryResource Pre-Invocation
00411 (0X019B) (OS) Dos32QueryResourceSize Post-Invocation
00412 (0X019C) (OS) DosDevLocl Pre-Invocation
00413 (0X019D) (OS) DosDevLocl Post-Invocation
00414 (0X019E) (OS) DosSetRelMaxFH Pre-Invocation
00415 (0X019F) (OS) DosSetRelMaxFH Post-Invocation
00416 (0X01A0) (OS) Dos32InitializePorthole Pre-Invocation
00417 (0X01A1) (OS) Dos32InitializePorthole Post-Invocation
00418 (0X01A2) (OS) Dos32QueryHeaderInfo Pre-Invocation
00419 (0X01A3) (OS) Dos32QueryHeaderInfo Post-Invocation
00420 (0X01A4) (OS) Dos32QueryProcType Pre-Invocation
00421 (0X01A5) (OS) Dos32QueryProcType Post-Invocation
00422 (0X01A6) (OS) DosOpen2Compt Pre-Invocation
00423 (0X01A7) (OS) DosOpen2Compt Post-Invocation
00424 (0X01A8) (OS) Dos32Open2Compt Pre-Invocation
00425 (0X01A9) (OS) Dos32Open2Compt Post-Invocation
00426 (0X01AA) (OS) Dos32SetFHState Pre-Invocation
00427 (0X01AB) (OS) Dos32SetFHState Post-Invocation
00430 (0X01AE) (OS) Dos32QUERYFHSTATE Pre-Invocation
00431 (0X01AF) (OS) Dos32QUERYFHSTATE Post-Invocation
00432 (0X01B0) (OS) Dos32IRead Pre-Invocation
00433 (0X01B1) (OS) Dos32IRead Post-Invocation
00434 (0X01B2) (OS) Dos32IWrite Pre-Invocation
00435 (0X01B3) (OS) Dos32WWrite Post-Invocation
00436 (0X01B4) (OS) Dos32DumpProcess Pre-Invocation
00437 (0X01B5) (OS) Dos32DumpProcess Post-Invocation
00438 (0X01B6) (OS) Dos32SuppressPopUps Pre-Invocation
00439 (0X01B7) (OS) Dos32SuppressPopUps Post-Invocation
00440 (0X01B8) (OS) Dos32KillThread Pre-Invocation
00441 (0X01B9) (OS) Dos32KillThread Post-Invocation
00442 (0X01BA) (OS) Dos32ProtectSetFHState Pre-Invocation
00443 (0X01BB) (OS) Dos32ProtectSetFHState Post-Invocation
00444 (0X01BC) (OS) Dos32PROTECTQUERYFHSTATE Pre-Invocation
00445 (0X01BD) (OS) Dos32PROTECTQUERYFHSTATE Post-Invocation
00446 (0X01BE) (OS) DosProtectChgFilePtr Pre-Invocation
00447 (0X01BF) (OS) DosProtectChgFilePtr Post-Invocation
00448 (0X01C0) (OS) DosProtectClose Pre-Invocation
00449 (0X01C1) (OS) DosProtectClose Post-Invocation
00450 (0X01C2) (OS) DosCloseChangeNotify Pre-Invocation
00451 (0X01C3) (OS) DosCloseChangeNotify Post-Invocation
00452 (0X01C4) (OS) DosProtectEnumAttribute Pre-Invocation
00453 (0X01C5) (OS) DosProtectEnumAttribute Post-Invocation
00454 (0X01C6) (OS) DosProtectFileIOPre-Invocation
00455 (0X01C7) (OS) DosProtectFileIOPost-Invocation
00456 (0X01C8) (OS) DosProtectFileLocks Pre-Invocation
00457 (0X01C9) (OS) DosProtectFileLocks Post-Invocation
00458 (0X01CA) (OS) DosForceDelete Pre-Invocation
00459 (0X01CB) (OS) DosForceDelete Post-Invocation
00460 (0X01CC) (OS) DosIPrProtectRead Pre-Invocation
00461 (0X01CD) (OS) DosIPrProtectRead Post-Invocation
00462 (0X01CE) (OS) DosIPrProtectSetFileInfo Pre-Invocation
00463 (0X01CF) (OS) DosIPrProtectSetFileInfo Post-Invocation
00464 (0X01D0) (OS) DosIPROTECTWrite Pre-Invocation
00465 (0X01D1) (OS) DosIPrProtectWrite Post-Invocation
00466 (0X01D2) (OS) DosProtectNewSize Pre-Invocation
00467 (0X01D3) (OS) DosProtectNewSize Post-Invocation
00468 (0X01D4) (OS) DOSPROTECTOPEN Pre-Invocation
00469 (0X01D5) (OS) DosProtectOpen Post-Invocation
00470 (0X01D6) (OS) DosOpenChangeNotify Pre-Invocation
00471 (0X01D7) (OS) DosOpenChangeNotify Post-Invocation
00472 (0X01D8) (OS) DosProtectQFHHandState Pre-Invocation
00473 (0X01D9) (OS) DosProtectQFHHandState Post-Invocation
00474 (0X01DA) (OS) DosProtectQFileInfo Pre-Invocation
00475 (0X01DB) (OS) DosProtectQFileInfo Post-Invocation
00476 (0X01DC) (OS) DosResetChangeNotify Pre-Invocation
00477 (0X01DD) (OS) DosResetChangeNotify Post-Invocation
00478 (0X01DE) (OS) DosProtectSetFHHandState Pre-Invocation
00479 (0X01DF) (OS) DosProtectSetFHHandState Post-Invocation
00480 (0X01E0) (OS) DosProtectSetFileInfo Pre-Invocation
00481 (0X01E1) (OS) DosProtectSetFileInfo Post-Invocation
00482 (0X01E2) (OS) Dos32PMPostEventSem Pre-Invocation
00483 (0X01E3) (OS) Dos32PMPostEventSem Post-Invocation
00484 (0X01E4) (OS) Dos32PMWaitEventSem Post-Invocation
00485 (0X01E5) (OS) Dos32PMWaitMuxWaitSem Pre-Invocation
00486 (0X01E6) (OS) Dos32PMWaitMuxWaitSem Post-Invocation
00487 (0X01E7) (OS) Dos32QueryExtLIBPATH Pre-Invocation
00488 (0X01E8) (OS) Dos32QueryExtLIBPATH Post-Invocation
00489 (0X01E9) (OS) Dos32SetExtLIBPATH Pre-Invocation
00490 (0X01EA) (OS) Dos32SetExtLIBPATH Post-Invocation
00491 (0X01EB) (OS) Dos32VERIFYFPIDTID Pre-Invocation
00492 (0X01EC) (OS) Dos32VERIFYFPIDTID Post-Invocation
00493 (0X01ED) (OS) Dos32PMWaitEventSem Pre-Invocation
00494 (0X01EE) (OS) Dos32CancelLockRequest Pre-Invocation
00495 (0X01EF) (OS) Dos32CancelLockRequest Post-Invocation
00496 (0X01F0) (OS) Dos32SetFileLocks Pre-Invocation
00497 (0X01F1) (OS) Dos32SetFileLocks Post-Invocation
00498 (0X01F2) (OS) Dos32ProtectSetFileLocks Pre-Invocation
00499 (0X01F3) (OS) Dos32ProtectSetFileLocks Post-Invocation
00500 (0X01F4) (OS) DosCreateSpinLock Pre-Invocation
00501 (0X01F5) (OS) DosCreateSpinLock Post-Invocation
00502 (0X01F6) (OS) DosAcquireSpinLock Pre-Invocation
00503 (0X01F7) (OS) DosAcquireSpinLock Post-Invocation
00504 (0X01F8) (OS) DosReleaseSpinLock Pre-Invocation
Trace Events for OS2KRNL Major Code: 0X0005, Sorted by Tracepoint

Allocate 00393 (0X0189)
DOSMUXSEMWAIT 00253 (0X00FD)
post2DOSENTERCRITSEC 00001 (0X0001)
post2DOSEXITCRITSEC 00002 (0X0002)
post2DOSHOLDSIGNAL 00003 (0X0003)
postAllocate 00394 (0X018A)
postDOS32ADDMUXWAITSEM 00370 (0X0172)
postDOS32ALIASMEM 00006 (0X0006)
postDOS32ALLOCMEM 00007 (0X0007)
postDOS32ALLOCPROTECTEDMEM 00008 (0X0008)
postDOS32ALLOCSHAREDMEM 00009 (0X0009)
postDOS32ASYNCTIMER 00378 (0X017A)
postDOS32CANCELCLOCKREQUEST 00495 (0X01EF)
postDOS32CLOSEEVENTSEM 00340 (0X0154)
postDOS32CLOSEMUTEXSEM 00354 (0X0162)
postDOS32CLOSEMUXWAITSEM 00366 (0X016E)
postDOS32CREATEEVENTSEM 00336 (0X0150)
postDOS32CREatemutexsem 00350 (0X015E)
postDOS32CREatemuxwaitsem 00362 (0X016A)
postDOS32CREATETHREAD 00010 (0X000A)
postDOS32DEBUG 00011 (0X000B)
postDOS32DELETEMUXWAITSEM 00372 (0X0174)
postDOS32DUMPPROCESS 00437 (0X01B5)
postDOS32EXITLIST 00012 (0X000C)
postDOS32FREEMEM 00013 (0X000D)
postDOS32FREERESOURCE 00332 (0X014C)
postDOS32GETNAMEDSHAREDMEM 00014 (0X000E)
postDOS32GETRESOURCE 00328 (0X0148)
postDOS32GETSHAREDMEM 00015 (0X000F)
postDOS32GIVESHAREDMEM 00016 (0X0010)
postDOS32ASYNCTIMER 00153 (0X0099)
postDOS32INITIALIZEPORThole 00417 (0X01A1)
postDOS32PROTECTQUERYFHOSTATE 00445 (0X01BD)
postDOS32PROTECTREAD 00059 (0X01FD)
postDOS32PROTECTSETFHSTATE 00443 (0X01BB)
postDOS32PROTECTWRITE 00511 (0X01FF)
postDOS32QUERYFHSTATE 00431 (0X01AF)
postDOS32READ 00433 (0X01B1)
postDOS32SETFHSTATE 00429 (0X01AD)
postDOS32STARTTIMER 00154 (0X009A)
postDOS32WRITE 00435 (0X01B3)
postDOS32KILLTHREA D 00441 (0X01B9)
postDOS32OPENEVENTSEM 00338 (0X0152)
postDOS32OPENMUTEXSEM 00352 (0X0160)
postDOS32OPENMUXWAITSEM 00364 (0X016C)
postDOS32PMPOSTEVENTSEM 00483 (0X01E3)
postDOS32PMWAITMUXWAITSEM 00484 (0X01E4)
postDOS32POSTEVENTSEM 00344 (0X0158)
postDOS32PROTECTSETFILELOCKS 00499 (0X01F3)
postDOS32QUERYEVENTSEM 0048 (0X015C)
OS2KRNL Major Code: 0X0005 Minor Code: 1 (0X0001)

Description
(OS) DosEnterCritSec Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSENTERCRITSEC

Minor Code
1 (0X0001)

Trace Groups
TK

Trace Types
POST

Traced Parameters

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 2 (0X0002)

Description
(OS) DosExitCritSec Post-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSEXITCRITSEC

Minor Code
2 (0X0002)

Trace Groups
TK

Trace Types
POST

Traced Parameters
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 3 (0X0003)

Description
(OS) DosHoldSignal Post-Invocation (2)

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSHOLDSIGNAL

Minor Code
3 (0X0003)

Trace Groups
SIG

Trace Types
POST

Traced Parameters
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 6 (0X0006)

Description
(OS) Dos32AliasMem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALIASMEM

Minor Code
6 (0X0006)

Trace Groups
VM

Trace Types
POST

Traced Parameters
OS2KRNL Major Code: 0X0005 Minor Code: 7 (0X0007)

**Description**
(OS) Dos32AllocMem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCMEM

**Minor Code**
7 (0X0007)

**Trace Groups**
VM

**Trace Types**
POST

**Traced Parameters**

Address = %F Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 8 (0X0008)

**Description**
(OS) Dos32AllocProtectedMem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCPROTECTEDMEM

**Minor Code**
8 (0X0008)

**Trace Groups**
VM

**Trace Types**
POST

**Traced Parameters**

Address = %F Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 9 (0X0009)

**Description**
(OS) Dos32AllocSharedMem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCSHAREDMEM

Minor Code
9 (0X0009)

Trace Groups
VM

Trace Types
POST

Traced Parameters
Address = %F Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 10 (0X000A)

Description
(OS) Dos32CreateThread Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATETHREAD

Minor Code
10 (0X000A)

Trace Groups
TK

Trace Types
POST

Traced Parameters
Thread ID = %D Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 11 (0X000B)

Description
(OS) Dos32Debug Dos Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DEBUG

Minor Code
11 (0X000B)

Trace Groups
TK

Trace Types
POST
**Traced Parameters**

Return code = %D

-----------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 12 (0X000C)**

**Description**
(OS) Dos32ExitList Dos Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32EXITLIST

**Minor Code**
12 (0X000C)

**Trace Groups**
TK

**Trace Types**
POST

**Traced Parameters**

Return code = %D

-----------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 13 (0X000D)**

**Description**
(OS) Dos32FreeMem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32FREEMEM

**Minor Code**
13 (0X000D)

**Trace Groups**
VM

**Trace Types**
POST

**Traced Parameters**

Return code = %D

-----------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 14 (0X000E)**
Description: (OS) Dos32GetNamedSharedMem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETNAMEDSHAREDMEM

Minor Code: 14 (0X000E)

Trace Groups: VM

Trace Types: POST

Traced Parameters:

Address = %F Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 15 (0X000F)

Description: (OS) Dos32GetSharedMem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETSHAREDMEM

Minor Code: 15 (0X000F)

Trace Groups: VM

Trace Types: POST

Traced Parameters:

Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 16 (0X0010)

Description: (OS) Dos32GiveSharedMem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GIVESHAREDMEM

Minor Code: 16 (0X0010)

Trace Groups: VM

Trace Types: POST
OS2KRNL Major Code: 0X0005 Minor Code: 17 (0X0011)

Description
(OS) Dos32QueryMem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMEM

Minor Code
17 (0X0011)

Trace Groups
VM

Trace Types
POST

Traced Parameters
Actual size = %D Flags = %D
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 18 (0X0012)

Description
(OS) Dos32QueryMemState Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMEMSTATE

Minor Code
18 (0X0012)

Trace Groups
VM

Trace Types
POST

Traced Parameters
Actual size = %D Flags = %D
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 19 (0X0013)
**Description**
(OS) Dos32SetMem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETMEM

**Minor Code**
19 (0X0013)

**Trace Groups**
VM

**Trace Types**
POST

**Traced Parameters**

Return code = %D

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 20 (0X0014)

**Description**
(OS) DosAllocHuge Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCHUGE

**Minor Code**
20 (0X0014)

**Trace Groups**
SEL

**Trace Types**
POST

**Traced Parameters**

Base selector = %W Return code = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 21 (0X0015)

**Description**
(OS) DosAllocProtHuge Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCPROTHUGE

**Minor Code**
21 (0X0015)

**Trace Groups**
SEL
Trace Types: POST

Traced Parameters:
Base selector = %W Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 22 (0X0016)

Description: (OS) DosAllocProtSeg Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCPROTSEG
Minor Code: 22 (0X0016)
Trace Groups: SEL
Trace Types: POST
Traced Parameters:
Selector = %W Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 23 (0X0017)

Description: (OS) DosAllocSeg Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCSEG
Minor Code: 23 (0X0017)
Trace Groups: SEL
Trace Types: POST
Traced Parameters:
Selector = %W Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 24 (0X0018)
**Description**  
(OS) DosAllocShrProtSeg Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCSHRPROTSEG

**Minor Code**  
24 (0X0018)

**Trace Groups**  
SEL

**Trace Types**  
POST

**Traced Parameters**  
Selector = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 25 (0X0019)

**Description**  
(OS) DosAllocShrSeg Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCSHRSEG

**Minor Code**  
25 (0X0019)

**Trace Groups**  
SEL

**Trace Types**  
POST

**Traced Parameters**  
Selector = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 26 (0X001A)

**Description**  
(OS) DosBeep Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSBEEP

**Minor Code**  
26 (0X001A)

**Trace Groups**  
IO
Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 27 (0X001B)

Description
(OS) DosBufReset Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSBUFRESET

Minor Code
27 (0X001B)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 28 (0X001C)

Description
(OS) DosCallNmPipe Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCALLNMPipe

Minor Code
28 (0X001C)

Trace Groups
PIP

Trace Types
POST

Traced Parameters
Bytes out = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 29 (0X001D)
**Description**  (OS) DosChDir Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCHDIR

**Minor Code**  29 (0X001D)

**Trace Groups**  FS

**Trace Types**  POST

**Traced Parameters**

Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 30 (0X001E)**

**Description**  (OS) DosChgFilePtr Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCHGFILEPTR

**Minor Code**  30 (0X001E)

**Trace Groups**  FS

**Trace Types**  POST

**Traced Parameters**

Location = %W%W Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 31 (0X001F)**

**Description**  (OS) DosCliAccess Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLIACCESS

**Minor Code**  31 (0X001F)

**Trace Groups**  TK
Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 32 (0X0020)

Description
(OS) DosClose Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSE

Minor Code
32 (0X0020)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 33 (0X0021)

Description
(OS) DosCloseSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSESEM

Minor Code
33 (0X0021)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Return value = %W

OS2KRNL Major Code: 0X0005 Minor Code: 34 (0X0022)
**Description**
(OS) DosConnectNmPipe Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCONNECTNPIPE

**Minor Code**
34 (0X0022)

**Trace Groups**
PIP

**Trace Types**
POST

**Traced Parameters**
Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 35 (0X0023)**

**Description**
(OS) DosCreateCSAlias Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATECSALIAS

**Minor Code**
35 (0X0023)

**Trace Groups**
SEL

**Trace Types**
POST

**Traced Parameters**
Code selector = %W Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 36 (0X0024)**

**Description**
(OS) DosCreateSem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATESEM

**Minor Code**
36 (0X0024)

**Trace Groups**
SEL
Trace Types: POST

Traced Parameters:
Handle = %W, Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 37 (0X0025)

Description: (OS) DosCreateThread Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATETHREAD
Minor Code: 37 (0X0025)
Trace Groups: TK
Trace Types: POST
Traced Parameters:
Thread ID = %W, Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 38 (0X0026)

Description: (OS) DosCWait Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCWAIT
Minor Code: 38 (0X0026)
Trace Groups: TK
Trace Types: POST
Traced Parameters:
Pid = %W, Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 39 (0X0027)
**Description**  
(OS) DosDelete Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDELETE

**Minor Code**  
39 (0X0027)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**

Return code = %W  

OS2KRNL Major Code: 0X0005 Minor Code: 40 (0X0028)

**Description**  
(OS) DosDevConfig Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVCONFIG

**Minor Code**  
40 (0X0028)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**

Return code = %W  

OS2KRNL Major Code: 0X0005 Minor Code: 41 (0X0029)

**Description**  
(OS) DosDevioctl2 Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVIOCTL2

**Minor Code**  
41 (0X0029)

**Trace Groups**  
FS
Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 42 (0X002A)

Description
(OS) DosDevIoctl Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVOCTL

Minor Code
42 (0X002A)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 43 (0X002B)

Description
(OS) DosDisConnectMmPipe Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDISCONNECTMMPipe

Minor Code
43 (0X002B)

Trace Groups
PIP

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 44 (0X002C)
Description: (OS) DosDupHandle Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDUPHANDLE

Minor Code: 44 (0X002C)

Trace Groups: FS

Trace Types: POST

Traced Parameters:

New handle = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 45 (0X002D)

Description: (OS) DosEditName Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEDITNAME

Minor Code: 45 (0X002D)

Trace Groups: FS

Trace Types: POST

Traced Parameters:

Resultant string = %S
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 47 (0X002F)

Description: (OS) DosError Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSERROR

Minor Code: 47 (0X002F)
Trace Groups: TK
Trace Types: POST
Traced Parameters:

Return value = %W

-------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 49 (0X0031)

Description: (OS) Dos32Exit Dos Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEXIT
Minor Code: 49 (0X0031)
Trace Groups: TK
Trace Types: POST
Traced Parameters:

Return code = %D

-------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 51 (0X0033)

Description: (OS) Dos32ExitList Dos Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEXITLIST
Minor Code: 51 (0X0033)
Trace Groups: TK
Trace Types: POST
Traced Parameters:

Return code = %D

-------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 52 (0X0034)

Description
(OS) DosFileIO Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFILEIO

Minor Code
52 (0X0034)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Return code = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 53 (0X0035)

Description
(OS) DosFileLocks Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFILELOCKS

Minor Code
53 (0X0035)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Return code = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 54 (0X0036)

Description
(OS) DosFindClose Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDCLOSE

Minor Code
54 (0X0036)
Trace Groups: FS
Trace Types: POST
Traced Parameters:

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 55 (0X0037)

Description: (OS) DosFindFirst2 Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFIRST2
Minor Code: 55 (0X0037)
Trace Groups: FS
Trace Types: POST
Traced Parameters:

Search handle = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 56 (0X0038)

Description: (OS) DosFindFirst Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFIRST
Minor Code: 56 (0X0038)
Trace Groups: FS
Trace Types: POST
Traced Parameters:

Search handle = %W Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 57 (0X0039)

Description
(OS) DosFindFromName Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFROMNAME

Minor Code
57 (0X0039)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Search count = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 58 (0X003A)

Description
(OS) DosFindNext Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNEXT

Minor Code
58 (0X003A)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 59 (0X003B)

Description
(OS) DosFindNotifyClose Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYCLOSE

Minor Code
OS2KRNL Major Code: 0X0005 Minor Code: 60 (0X003C)

Description
(OS) DosFindNotifyFirst Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYFIRST

Minor Code
60 (0X003C)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Search count = %W Handle = %W
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 61 (0X003D)

Description
(OS) DosFindNotifyNext Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYNEXT

Minor Code
61 (0X003D)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Change count = %W Return code = %W
OS2KRNL Major Code: 0X0005 Minor Code: 62 (0X003E)

**Description**
(OS) DosFlagProcess Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFLAGPROCESS

**Minor Code**
62 (0X003E)

**Trace Groups**
SIG

**Trace Types**
POST

**Traced Parameters**
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 63 (0X003F)

**Description**
(OS) DosFreeModule Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREEMODULE

**Minor Code**
63 (0X003F)

**Trace Groups**
LDR

**Trace Types**
POST

**Traced Parameters**
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 64 (0X0040)

**Description**
(OS) DosFreeSeg Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREESEG

**Minor Code**
64 (0X0040)

**Trace Groups**
SEL

**Trace Types**
POST

**Traced Parameters**
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 65 (0X0041)

**Description**
(OS) DosFSAttach Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFSATTACH

**Minor Code**
65 (0X0041)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 66 (0X0042)

**Description**
(OS) DosFSCtl Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFSCTL

**Minor Code**
66 (0X0042)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 67 (0X0043)

Description
(OS) DosGetDateTime Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETDATETIME

Minor Code
67 (0X0043)

Trace Groups
TIM

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 68 (0X0044)

Description
(OS) DosGetModHandle Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETMODHANDLE

Minor Code
68 (0X0044)

Trace Groups
LDR

Trace Types
POST

Traced Parameters
Module handle = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 69 (0X0045)

Description
(OS) DosGetModName Post-Invocation
Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETMODNAME

Minor Code  
69 (0X0045)

Trace Groups  
LDR

Trace Types  
POST

Traced Parameters

Module name = %S
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 70 (0X0046)

Description  
(OS) DosGetPid Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPID

Minor Code  
70 (0X0046)

Trace Groups  
TK

Trace Types  
POST

Traced Parameters

PID = %W TID = %W PPID = %W

OS2KRNL Major Code: 0X0005 Minor Code: 71 (0X0047)

Description  
(OS) DosGetProcAddr Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPROCADDR

Minor Code  
71 (0X0047)

Trace Groups  
LDR

Trace Types  
POST
Traced Parameters

Proc addr = %W:%W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 72 (0X0048)

Description
(OS) DosGetPrty Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPRTY

Minor Code
72 (0X0048)

Trace Groups
TK

Trace Types
POST

Traced Parameters

Priority = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 73 (0X0049)

Description
(OS) DosGetResource Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETRESOURCE

Minor Code
73 (0X0049)

Trace Groups
LDR

Trace Types
POST

Traced Parameters

Selector = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 74 (0X004A)
(OS) DosGetSeg Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETSEG

Minor Code: 74 (0X004A)
Trace Groups: SEL
Trace Types: POST

Traced Parameters:
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 75 (0X004B)

(OS) DosGetShrSeg Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETSHRSEG

Minor Code: 75 (0X004B)
Trace Groups: SEL
Trace Types: POST

Traced Parameters:
Selector = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 76 (0X004C)

(OS) DosGetVersion Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETVERSION

Minor Code: 76 (0X004C)
Trace Groups: TK
Trace Types: POST
Traced Parameters

Major/minor version = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 77 (0X004D)

Description
(OS) DosGiveSeg Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGIVESEG

Minor Code
77 (0X004D)

Trace Groups
SEL

Trace Types
POST

Traced Parameters
Selector = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 78 (0X004E)

Description
(OS) DosHoldSignal Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSHOLDSEGSIGNAL

Minor Code
78 (0X004E)

Trace Groups
SIG

Trace Types
POST

Traced Parameters
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 79 (0X004F)
(OS) DosICopy Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSICOPY

Minor Code 79 (0X004F)

Trace Groups FS

Trace Types POST

Traced Parameters

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 80 (0X0050)

(OS) DosIExecPgm Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIEXECPGM

Minor Code 80 (0X0050)

Trace Groups TK

Trace Types POST

Traced Parameters

Pid/Term Code = %W Result Code = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 83 (0X0053)

(OS) DosISetCP Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETCP

Minor Code 83 (0X0053)

Trace Groups TK
OS2KRNL Major Code: 0X0005 Minor Code: 84 (0X0054)

Description  (OS) DosKillProcess Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSKILLPROCESS
Minor Code  84 (0X0054)
Trace Groups  SIG
Trace Types  POST
Traced Parameters

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 85 (0X0055)

Description  (OS) DosLoadModule Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSLOADMODULE
Minor Code  85 (0X0055)
Trace Groups  LDR
Trace Types  POST
Traced Parameters

Module handle = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 86 (0X0056)
**Description**  
(OS) DosMakeNmPipe Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMAKENMPIPE

**Minor Code**  
86 (0X0056)

**Trace Groups**  
PIP

**Trace Types**  
POST

**Traced Parameters**
Handle = %W Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 87 (0X0057)**

**Description**  
(OS) DosMakePipe Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMAKEPIPE

**Minor Code**  
87 (0X0057)

**Trace Groups**  
PIP

**Trace Types**  
POST

**Traced Parameters**
Read handle = %W Write handle = %W
Return code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 88 (0X0058)**

**Description**  
(OS) DosMkDir2 Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMKDIR2

**Minor Code**  
88 (0X0058)
OS2KRNL Major Code: 0X0005 Minor Code: 89 (0X0059)

Description  
(OS) DosMkDir Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMKDIR

Minor Code  
89 (0X0059)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 95 (0X005F)

Description  
(OS) DosMove Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMOVE

Minor Code  
95 (0X005F)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 96 (0X0060)

Description  
(OS) DosMuxSemWait Post-Invocation (Waited)

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMUXSEMWAIT

Minor Code  
96 (0X0060)

Trace Groups  
SEM

Trace Types  
PRE

Traced Parameters  
Index = %W Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 97 (0X0061)

Description  
(OS) DosNewSize Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSNEWSIZE

Minor Code  
97 (0X0061)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 98 (0X0062)

Description  
(OS) DosOpen2 Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN2

Minor Code  
98 (0X0062)
**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Action = %W Handle = %W
Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 99 (0X0063)**

**Description**
(OS) DosOpen Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN

**Minor Code**
99 (0X0063)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Action = %W Handle = %W
Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 100 (0X0064)**

**Description**
(OS) DosOpenSem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPENSEM

**Minor Code**
100 (0X0064)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**
Description: (OS) DosOpLockRelease Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPLOCKRELEASE

Minor Code
101 (0X0065)

Trace Groups
FS

Trace Types
POST

Traced Parameters
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 102 (0X0066)

Description: (OS) DosOpLockWait Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPLOCKWAIT

Minor Code
102 (0X0066)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Server Key = %D Kernel Key = %D

OS2KRNL Major Code: 0X0005 Minor Code: 103 (0X0067)

Description: (OS) DosPeekNmPipe Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPEEKNMPIPE
Minor Code: 103 (0x0067)
Trace Groups: PIP
Trace Types: POST
Traced Parameters:
Bytes read = %W
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 104 (0x0068)

Description: (OS) DosPhysicalDisk Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPHYSICALDISK
Minor Code: 104 (0x0068)
Trace Groups: FS
Trace Types: POST
Traced Parameters:
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 105 (0x0069)

Description: (OS) DosQNmPipeState Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMPIPESEMSTATE
Minor Code: 105 (0x0069)
Trace Groups: PIP
Trace Types: POST
Traced Parameters:
Return code = %W
OS2KRNL Major Code: 0X0005 Minor Code: 106 (0X006A)

Description  
(OS) DosPortAccess Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPORTACCESS

Minor Code  
106 (0X006A)

Trace Groups  
TK

Trace Types  
POST

Traced Parameters  
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 108 (0X006C)

Description  
(OS) DosQCurDir Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQCURDIR

Minor Code  
108 (0X006C)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
Current Dir = %S
Len = %W Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 109 (0X006D)

Description  
(OS) DosQCurDisk Post-Invocation
Tracepoint

OS2KRNL Major Code: 0X0005 Minor Code: 110 (0X006E)

Description
(OS) DosQFHandState Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFHANDSTATE

Minor Code
110 (0X006E)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Handle State = %W Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 111 (0X006F)

Description
(OS) DosQFileInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFILEINFO

Minor Code
111 (0X006F)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Default Drive = %W Logical Map = %D
Return Code = %W
Traced Parameters

Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0x0005 Minor Code: 112 (0x0070)

Description
(OS) DosQFileMode Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFILEMODE

Minor Code
112 (0X0070)

Trace Groups
FS

Trace Types
POST

Traced Parameters

File Mode = %W Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0x0005 Minor Code: 113 (0x0071)

Description
(OS) DosQFSAttach Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFSATTACH

Minor Code
113 (0X0071)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0x0005 Minor Code: 114 (0x0072)
**Description**
(OS) DosQFSInfo Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFSINFO

**Minor Code**
114 (0X0072)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 115 (0X0073)**

**Description**
(OS) DosQHandType Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQHANDTYPE

**Minor Code**
115 (0X0073)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Handle Type = %W Flags = %W
Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 116 (0X0074)**

**Description**
(OS) DosQNmPHandState Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMMPHANDSTATE

**Minor Code**
116 (0X0074)

**Trace Groups**
P2P
OS2KRNL Major Code: 0X0005 Minor Code: 117 (0X0075)

Description: (OS) DosQNmPipeInfo Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMPIPEINFO
Minor Code: 117 (0X0075)
Trace Groups: PIP
Trace Types: POST
Traced Parameters
Handle state = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 118 (0X0076)

Description: (OS) DosQPathInfo Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQPATHINFO
Minor Code: 118 (0X0076)
Trace Groups: FS
Trace Types: POST
Traced Parameters
Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 119 (0X0077)
Description: (OS) DosQSysInfo Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQSYSINFO

Minor Code: 119 (0X0077)

Trace Groups: FS

Trace Types: POST

Traced Parameters: Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 120 (0X0078)

Description: (OS) DosQVerify Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQVERIFY

Minor Code: 120 (0X0078)

Trace Groups: FS

Trace Types: POST

Traced Parameters: Verify Flag = %W Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 121 (0X0079)

Description: (OS) DosRawReadNmPipe Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRAWREADNMPPIPE

Minor Code: 121 (0X0079)

Trace Groups: PIP
Trace Types
- POST

Traced Parameters
- Bytes read = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 122 (0X007A)

Description
- (OS) DosRawWriteNmPipe Post-Invocation

Tracepoint
- Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRAWWRITENMPIPE

Minor Code
- 122 (0X007A)

Trace Groups
- PIP

Trace Types
- POST

Traced Parameters
- Bytes written = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 123 (0X007B)

Description
- (OS) DosIRead Post-Invocation

Tracepoint
- Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIREAD

Minor Code
- 123 (0X007B)

Trace Groups
- FS

Trace Types
- POST

Traced Parameters
- Bytes Read = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 125 (0X007D)
Description: (OS) DosReallocHuge Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSREALLOCHUGE

Minor Code: 125 (0X007D)

Trace Groups: SEL

Trace Types: POST

Traced Parameters:

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 126 (0X007E)

Description: (OS) DosReallocSeg Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSREALLOCSEG

Minor Code: 126 (0X007E)

Trace Groups: SEL

Trace Types: POST

Traced Parameters:

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 127 (0X007F)

Description: (OS) Dos32ResumeThread Dos Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRESUMETHREAD

Minor Code: 127 (0X007F)

Trace Groups: TK
Trace Types
POST

Traced Parameters

Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 128 (0X0080)

Description
(OS) DosRmDir Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRMDIR

Minor Code
128 (0X0080)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 129 (0X0081)

Description
(OS) DosSelectDisk Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSELECTDISK

Minor Code
129 (0X0081)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 133 (0X0085)
Description: (OS) DosSemSetWait Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSEMSETWAIT

Minor Code: 133 (0X0085)

Trace Groups: SEM

Trace Types: POST

Traced Parameters:

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 135 (0X0087)

Description: (OS) DosSendSignal Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSENDSIGNAL

Minor Code: 135 (0X0087)

Trace Groups: SIG

Trace Types: POST

Traced Parameters:

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 137 (0X0089)

Description: (OS) DosSetDateTime Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETDATETIME

Minor Code: 137 (0X0089)

Trace Groups: TIM
Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 138 (0X008A)

Description
(OS) DosSetFHandState Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFHANDSTATE

Minor Code
138 (0X008A)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 139 (0X008B)

Description
(OS) DosSetFileInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFILEINFO

Minor Code
139 (0X008B)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 140 (0X008C)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSetFileMode Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFILEMODE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>140 (0X008C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %W</td>
</tr>
</tbody>
</table>

OS2KRNL Major Code: 0X0005 Minor Code: 141 (0X008D)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSetFSInfo Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFSINFO</td>
</tr>
<tr>
<td>Minor Code</td>
<td>141 (0X008D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %W</td>
</tr>
</tbody>
</table>

OS2KRNL Major Code: 0X0005 Minor Code: 142 (0X008E)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSetMaxFH Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETMAXFH</td>
</tr>
<tr>
<td>Minor Code</td>
<td>142 (0X008E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
</tbody>
</table>
Trace Types  
POST

Traced Parameters  
Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 143 (0X008F)

Description  
(OS) DosSetNmPHandState Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETNMPHANDSTATE

Minor Code  
143 (0X008F)

Trace Groups  
PIP

Trace Types  
POST

Traced Parameters  
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 144 (0X0090)

Description  
(OS) DosSetNmPipeSem Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETNMPIPESEM

Minor Code  
144 (0X0090)

Trace Groups  
PIP

Trace Types  
POST

Traced Parameters  
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 145 (0X0091)
Description: (OS) DosSetPathInfo Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETPATHINFO

Minor Code: 145 (0X0091)

Trace Groups: FS

Trace Types: POST

Traced Parameters:

Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 146 (0X0092)

Description: (OS) DosSetPrty Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETPRTY

Minor Code: 146 (0X0092)

Trace Groups: TK

Trace Types: POST

Traced Parameters:

Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 147 (0X0093)

Description: (OS) DosSetSigHandler Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETSIGHANDLER

Minor Code: 147 (0X0093)

Trace Groups: SIG
Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 148 (0X0094)

Description
(OS) DosSetVec Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETVEC

Minor Code
148 (0X0094)

Trace Groups
SIG

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 149 (0X0095)

Description
(OS) DosSetVerify Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETVERIFY

Minor Code
149 (0X0095)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 150 (0X0096)
**Description**  
(OS) DosSleep Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSLEEP

**Minor Code**  
150 (0X0096)

**Trace Groups**  
TK

**Trace Types**  
POST

**Traced Parameters**  
Return code = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 151 (0X0097)**

**Description**  
(OS) Dos32SuspendThread Dos Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSUSPENDTHREAD

**Minor Code**  
151 (0X0097)

**Trace Groups**  
TK

**Trace Types**  
POST

**Traced Parameters**  
Return code = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 152 (0X0098)**

**Description**  
(OS) DosSystemService Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSYSTEMSERVICE

**Minor Code**  
152 (0X0098)

**Trace Groups**  
TK
OS2KRNL Major Code: 0X0005 Minor Code: 153 (0X0099)

Description
(OS) DosTimerAsync Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IASYNCTIMER

Minor Code
153 (0X0099)

Trace Groups
TIM

Trace Types
POST

Traced Parameters
Return code = %W Handle = %W

-----------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 154 (0X009A)

Description
(OS) DosTimerStart Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ISTARTTIMER

Minor Code
154 (0X009A)

Trace Groups
TIM

Trace Types
POST

Traced Parameters
Return code = %W Handle = %W

-----------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 155 (0X009B)
Description: (OS) Dos32StopTimer Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32STOPTIMER

Minor Code: 155 (0X009B)

Trace Groups: TIM

Trace Types: POST

Traced Parameters:

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 156 (0X009C)

Description: (OS) DosTransactNmPipe Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSTRANSACTNPIPE

Minor Code: 156 (0X009C)

Trace Groups: PIP

Trace Types: POST

Traced Parameters:

Bytes out = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 157 (0X009D)

Description: (OS) DosWaitNmPipe Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSWAITNPIPE

Minor Code: 157 (0X009D)

Trace Groups: PIP
Trace Types
POST

Traced Parameters

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 158 (0X009E)

Description
(OS) DosIWrite Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIWRITE

Minor Code
158 (0X009E)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Bytes Written = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 163 (0X00A3)

Description
(OS) Dos32AliasMem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALIASMEM

Minor Code
163 (0X00A3)

Trace Groups
VM

Trace Types
PRE

Traced Parameters

Address = %F Size = %D

OS2KRNL Major Code: 0X0005 Minor Code: 164 (0X00A4)
Description: (OS) Dos32AllocMem Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCMEM
Minor Code: 164 (0X00A4)
Trace Groups: VM
Trace Types: PRE
Traced Parameters:
  Address = %F Size = %D
  Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 165 (0X00A5)

Description: (OS) Dos32AllocProtectedMem Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCPROTECTEDMEM
Minor Code: 165 (0X00A5)
Trace Groups: VM
Trace Types: PRE
Traced Parameters:
  Size = %D Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 166 (0X00A6)

Description: (OS) Dos32AllocSharedMem Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCSHAREDMEM
Minor Code: 166 (0X00A6)
OS2KRNL Major Code: 0X0005 Minor Code: 167 (0X00A7)

Description
(OS) Dos32CreateThread Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATETHREAD

Minor Code
167 (0X00A7)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
Stack size = %D Flags = %D
Arg pointer = %F Starting EIP = %F

OS2KRNL Major Code: 0X0005 Minor Code: 168 (0X00A8)

Description
(OS) Dos32Debug Dos Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DEBUG

Minor Code
168 (0X00A8)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
No parameters traced.
OS2KRNL Major Code: 0X0005 Minor Code: 169 (0X00A9)

**Description**
(OS) Dos32ExitList Dos Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32EXITLIST

**Minor Code**
169 (0X00A9)

**Trace Groups**
TK

**Trace Types**
PRE

**Traced Parameters**
Function = %D  Address = %F

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 170 (0X00AA)

**Description**
(OS) Dos32FreeMem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32FREEMEM

**Minor Code**
170 (0X00AA)

**Trace Groups**
VM

**Trace Types**
PRE

**Traced Parameters**
Address = %F

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 171 (0X00AB)

**Description**
(OS) Dos32GetNamedSharedMem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETNAMEDSHAREDMEM

**Minor Code**
171 (0X00AB)
Trace Groups
VM
Trace Types
PRE
Traced Parameters
Name = %S
Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 172 (0X00AC)

Description
(OS) Dos32GetSharedMem Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETSHAREDMEM
Minor Code
172 (0X00AC)
Trace Groups
VM
Trace Types
PRE
Traced Parameters
Address = %F Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 173 (0X00AD)

Description
(OS) Dos32GiveSharedMem Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GIVESHAREDMEM
Minor Code
173 (0X00AD)
Trace Groups
VM
Trace Types
PRE
Traced Parameters
Address = %F Process ID = %D
OS2KRNL Major Code: 0X0005 Minor Code: 174 (0X00AE)

Description:
(OS) Dos32QueryMem Pre-Invocation

Tracepoint:
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMEM

Minor Code:
174 (0X00AE)

Trace Groups:
VM

Trace Types:
PRE

Traced Parameters:
Address = %F Size = %D

OS2KRNL Major Code: 0X0005 Minor Code: 175 (0X00AF)

Description:
(OS) Dos32QueryMemState Pre-Invocation

Tracepoint:
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMEMSTATE

Minor Code:
175 (0X00AF)

Trace Groups:
VM

Trace Types:
PRE

Traced Parameters:
Address = %F Size = %D

OS2KRNL Major Code: 0X0005 Minor Code: 176 (0X00B0)

Description:
(OS) Dos32SetMem Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETMEM

Minor Code
176 (0X00B0)

Trace Groups
VM

Trace Types
PRE

Traced Parameters
Address = %F Size = %D
Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 177 (0X00B1)

Description
(OS) DosAllocHuge Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCHUGE

Minor Code
177 (0X00B1)

Trace Groups
SEL

Trace Types
PRE

Traced Parameters
Initial size=%W %W Max size = %W0000

OS2KRNL Major Code: 0X0005 Minor Code: 178 (0X00B2)

Description
(OS) DosAllocProtHuge Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCPROTHUGE

Minor Code
178 (0X00B2)

Trace Groups
SEL

Trace Types
PRE
Traced Parameters

Initial size=%W Max size = %W0000

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 179 (0X00B3)

Description (OS) DosAllocProtSeg Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCPROTSEG
Minor Code 179 (0X00B3)
Trace Groups SEL
Trace Types PRE
Traced Parameters

Flags=%W Size=%W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 180 (0X00B4)

Description (OS) DosAllocSeg Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCSEG
Minor Code 180 (0X00B4)
Trace Groups SEL
Trace Types PRE
Traced Parameters

Flags=%W Size=%W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 181 (0X00B5)
Description  (OS) DosAllocShrProtSeg Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCSHRPROTSEG
Minor Code  181 (0X00B5)
Trace Groups  SEL
Trace Types  PRE
Traced Parameters

Name = %S
Size = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 182 (0X00B6)

Description  (OS) DosAllocShrSeg Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCSHRSEG
Minor Code  182 (0X00B6)
Trace Groups  SEL
Trace Types  PRE
Traced Parameters

Name = %S
Size = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 183 (0X00B7)

Description  (OS) DosBeep Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSBEEP
Minor Code  183 (0X00B7)
Trace Groups  SEL
OS2KRNL Major Code: 0X0005 Minor Code: 184 (0X00B8)

Description
(OS) DosBufReset Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSBUFRESET

Minor Code
184 (0X00B8)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 185 (0X00B9)

Description
(OS) DosCallNmPipe Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCALLNMPipe

Minor Code
185 (0X00B9)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters
Name = %S

OS2KRNL Major Code: 0X0005 Minor Code: 186 (0X00BA)
**Description**  
(OS) DosChDir Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCHDIR

**Minor Code**  
186 (0X00BA)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**

Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 187 (0X00BB)

**Description**  
(OS) DosChgFilePtr Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCHGFILEPTR

**Minor Code**  
187 (0X00BB)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**

Type = %W Distance = %W%W
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 188 (0X00BC)

**Description**  
(OS) DosCliAccess Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLIAccess

**Minor Code**  
188 (0X00BC)
Trace Groups: TK
Trace Types: PRE
Traced Parameters: No parameters traced.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 189 (0X00BD)

Description: (OS) DosClose Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSE
Minor Code: 189 (0X00BD)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 190 (0X00BE)

Description: (OS) DosCloseSem Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSESEM
Minor Code: 190 (0X00BE)
Trace Groups: SEM
Trace Types: PRE
Traced Parameters: Semaphore handle = %W%W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 191 (0X00BF)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosConnectNmPipe Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCONNECTNMPipe</td>
</tr>
<tr>
<td>Minor Code</td>
<td>191 (0X00BF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PIP</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Handle = %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 192 (0X00C0)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosCreateCSAlias Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATECSALIAS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>192 (0X00C0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SEL</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Data selector = %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 193 (0X00C1)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosCreateSem Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATESEM</td>
</tr>
<tr>
<td>Minor Code</td>
<td>193 (0X00C1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SEM</td>
</tr>
</tbody>
</table>
Trace Types
  PRE

Traced Parameters

  Name = %S

-------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 194 (0X00C2)

Description
  (OS) DosCreateThread Pre-Invocation

Tracepoint
  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATETHREAD

Minor Code
  194 (0X00C2)

Trace Groups
  TK

Trace Types
  PRE

Traced Parameters

  Transfer Address = %A Stack Address = %A

-------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 195 (0X00C3)

Description
  (OS) DosCWait Pre-Invocation

Tracepoint
  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCWAIT

Minor Code
  195 (0X00C3)

Trace Groups
  TK

Trace Types
  PRE

Traced Parameters

  Pid = %W Wait = %W Action = %W

-------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 196 (0X00C4)
Description: (OS) DosDelete Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDELETE

Minor Code: 196 (0X00C4)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:
Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 197 (0X00C5)

Description: (OS) DosDevConfig Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVCONFIG

Minor Code: 197 (0X00C5)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:
Parm = %W Item = %W

OS2KRNL Major Code: 0X0005 Minor Code: 198 (0X00C6)

Description: (OS) DosDevioctl2 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVIOCTL2

Minor Code: 198 (0X00C6)

Trace Groups: FS
Trace Types
PRE

Traced Parameters
Handle = %W Category = %W Function = %W

OS2KRNL Major Code: 0X0005 Minor Code: 199 (0X00C7)

Description
(OS) DosDevIoctl Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVIOCTL

Minor Code
199 (0X00C7)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Handle = %W Category = %W Function = %W

OS2KRNL Major Code: 0X0005 Minor Code: 200 (0X00C8)

Description
(OS) DosDisConnectMmPipe Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDISCONNECTMMPipe

Minor Code
200 (0X00C8)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 201 (0X00C9)
**OS2KRNL Major Code: 0X0005 Minor Code: 201 (0X00C9)**

**Description**  
(OS) DosDupHandle Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDUPHANDLE

**Minor Code**  
201 (0X00C9)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
Handle = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 202 (0X00CA)**

**Description**  
(OS) DosEditName Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEDITNAME

**Minor Code**  
202 (0X00CA)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
Edit string = %S
Source string = %S
Level = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 203 (0X00CB)**

**Description**  
(OS) DosEnterCritSec Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSENTERCRITSEC

**Minor Code**
203 (0X00CB)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 204 (0X00CC)**

**Description**

(OS) DosError Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSERROR

**Minor Code**

204 (0X00CC)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Error setting = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 206 (0X00CE)**

**Description**

(OS) DosExit Dos Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEXIT

**Minor Code**

206 (0X00CE)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Function = %W ResultCode = %W

--------------------------------------------
<table>
<thead>
<tr>
<th>Major Code: 0X0005 Minor Code: 207 (0X00CF)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Code: 0X0005 Minor Code: 208 (0X00D0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Code: 0X0005 Minor Code: 209 (0X00D1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
</tbody>
</table>
FS

Trace Types  PRE

Traced Parameters

Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 210 (0X00D2)

Description  (OS) DosFileLocks Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFILELOCKS

Minor Code  210 (0X00D2)

Trace Groups  FS

Trace Types  PRE

Traced Parameters

Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 211 (0X00D3)

Description  (OS) DosFindClose Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDCLOSE

Minor Code  211 (0X00D3)

Trace Groups  FS

Trace Types  PRE

Traced Parameters

Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 212 (0X00D4)
**Description**
(OS) DosFindFirst2 Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFIRST2

**Minor Code**
212 (0X00D4)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 213 (0X00D5)

**Description**
(OS) DosFindFirst Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFIRST

**Minor Code**
213 (0X00D5)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 214 (0X00D6)

**Description**
(OS) DosFindFromName Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFROMNAME

**Minor Code**
214 (0X00D6)

**Trace Groups**
FS
Trace Types
PRE

Traced Parameters

Name = %S
Position = %F Directory Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 215 (0X00D7)

Description
(OS) DosFindNext Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNEXT

Minor Code
215 (0X00D7)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 216 (0X00D8)

Description
(OS) DosFindNotifyClose Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYCLOSE

Minor Code
216 (0X00D8)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

Handle = %W
OS2KRNL Major Code: 0X0005 Minor Code: 217 (0X00D9)

**Description**
(OS) DosFindNotifyFirst Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYFIRST

**Minor Code**
217 (0X00D9)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**

Name = %S
Attribute = %W Level = %W
Timeout = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 218 (0X00DA)

**Description**
(OS) DosFindNotifyNext Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYNEXT

**Minor Code**
218 (0X00DA)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**

Change count = %W Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 219 (0X00DB)

**Description**
(OS) DosFlagProcess Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFLAGPROCESS

Minor Code
219 (0X00DB)

Trace Groups
SIG

Trace Types
PRE

Traced Parameters

Pid = %W Action = %W
Signal = %W Arg = %W

OS2KRNL Major Code: 0X0005 Minor Code: 220 (0X00DC)

Description
(OS) DosFreeModule Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREEMODULE

Minor Code
220 (0X00DC)

Trace Groups
LDR

Trace Types
PRE

Traced Parameters

Module Handle = %F

OS2KRNL Major Code: 0X0005 Minor Code: 221 (0X00DD)

Description
(OS) DosFreeSeg Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREESEG

Minor Code
221 (0X00DD)

Trace Groups
SEL

Trace Types
PRE

Traced Parameters
OS2KRNL Major Code: 0X0005 Minor Code: 222 (0X00DE)

Description: (OS) DosFSAttach Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFSATTACH
Minor Code: 222 (0X00DE)
Trace Groups: FS
Trace Types: PRE
Traced Parameters:
  Device = %S
  FSD = %S
  OpFlag = %W

OS2KRNL Major Code: 0X0005 Minor Code: 223 (0X00DF)

Description: (OS) DosFSCtl Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFSCTL
Minor Code: 223 (0X00DF)
Trace Groups: FS
Trace Types: PRE
Traced Parameters:
  FSD = %S
  Handle = %W Route = %W

OS2KRNL Major Code: 0X0005 Minor Code: 224 (0X00E0)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosGetDateTime Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETDATETIME</td>
</tr>
<tr>
<td>Minor Code</td>
<td>224 (0X00E0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>TIM</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DateTime structure at %W%W</td>
</tr>
</tbody>
</table>

OS2KRNL Major Code: 0X0005 Minor Code: 225 (0X00E1)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosGetModHandle Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETMODHANDLE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>225 (0X00E1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>LDR</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Module name = %S</td>
</tr>
</tbody>
</table>

OS2KRNL Major Code: 0X0005 Minor Code: 226 (0X00E2)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosGetModName Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETMODNAME</td>
</tr>
<tr>
<td>Minor Code</td>
<td>226 (0X00E2)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>LDR</td>
</tr>
</tbody>
</table>
Trace Types
PRE

Traced Parameters
Module handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 227 (0X00E3)

Description
(OS) DosGetPid Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPID

Minor Code
227 (0X00E3)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 228 (0X00E4)

Description
(OS) DosGetProcAddr Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPROCADDR

Minor Code
228 (0X00E4)

Trace Groups
LDR

Trace Types
PRE

Traced Parameters
Module handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 229 (0X00E5)
**Description**
(OS) DosGetPrty Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPRTY

**Minor Code**
229 (0X00E5)

**Trace Groups**
TK

**Trace Types**
PRE

**Traced Parameters**

ID = %W Scope = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 230 (0X00E6)**

**Description**
(OS) DosGetResource Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETRESOURCE

**Minor Code**
230 (0X00E6)

**Trace Groups**
LDR

**Trace Types**
PRE

**Traced Parameters**

NameID = %W Type = %W
Module handle = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 231 (0X00E7)**

**Description**
(OS) DosGetSeg Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETSEG

**Minor Code**
231 (0X00E7)

**Trace Groups**
SEL

Trace Types
PRE

Traced Parameters
Selector = %W

OS2KRNL Major Code: 0X0005 Minor Code: 232 (0X00E8)

Description
(OS) DosGetShrSeg Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETSHRSEG

Minor Code
232 (0X00E8)

Trace Groups
SEL

Trace Types
PRE

Traced Parameters
Name = %S

OS2KRNL Major Code: 0X0005 Minor Code: 233 (0X00E9)

Description
(OS) DosGetVersion Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETVERSION

Minor Code
233 (0X00E9)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 234 (0X00EA)
**Description**  
(OS) DosGiveSeg Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGIVESEG

**Minor Code**  
234 (0X00EA)

**Trace Groups**  
SEL

**Trace Types**  
PRE

**Traced Parameters**

Selector = %W Pid = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 235 (0X00EB)

**Description**  
(OS) DosHoldSignal Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSHOLD SIGNAL

**Minor Code**  
235 (0X00EB)

**Trace Groups**  
SIG

**Trace Types**  
PRE

**Traced Parameters**

Action = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 236 (0X00EC)

**Description**  
(OS) DosICopy Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSICOPY

**Minor Code**  
236 (0X00EC)

**Trace Groups**  
FS
Trace Types
PRE

Traced Parameters
Source = %S
Target = %S
OpMode = %W

OS2KRNL Major Code: 0X0005 Minor Code: 237 (0X00ED)

Description
(OS) DosIExecPgm Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIEXECPGM

Minor Code
237 (0X00ED)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
Program name = %S
Exec flag = %W

OS2KRNL Major Code: 0X0005 Minor Code: 240 (0X00F0)

Description
(OS) DosISetCP Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETCP

Minor Code
240 (0X00F0)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
Codepage ID = %W
OS2KRNL Major Code: 0X0005 Minor Code: 241 (0X00F1)

**Description**
(OS) DosKillProcess Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSKILLPROCESS

**Minor Code**
241 (0X00F1)

**Trace Groups**
SIG

**Trace Types**
PRE

**Traced Parameters**

PID = %W

OS2KRNL Major Code: 0X0005 Minor Code: 242 (0X00F2)

**Description**
(OS) DosLoadModule Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSLOADMODULE

**Minor Code**
242 (0X00F2)

**Trace Groups**
LDR

**Trace Types**
PRE

**Traced Parameters**

Module name = %S

OS2KRNL Major Code: 0X0005 Minor Code: 243 (0X00F3)

**Description**
(OS) DosMakeNmPipe Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMAKENMPIPE

**Minor Code**
243 (0X00F3)

**Trace Groups**
PIP

**Trace Types**
PRE

**Traced Parameters**
Name = %S

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 244 (0X00F4)

**Description**
(OS) DosMakePipe Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMAKEPIPE

**Minor Code**
244 (0X00F4)

**Trace Groups**
PIP

**Trace Types**
PRE

**Traced Parameters**
Size of pipe = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 245 (0X00F5)

**Description**
(OS) DosMkDir2 Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMAKEPIPE

**Minor Code**
245 (0X00F5)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
OS2KRNL Major Code: 0X0005 Minor Code: 246 (0X00F6)

Description
(OS) DosMkDir Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMKDIR

Minor Code
246 (0X00F6)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 252 (0X00FC)

Description
(OS) DosMove Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMOVE

Minor Code
252 (0X00FC)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
New name = %S
Old name = %S

OS2KRNL Major Code: 0X0005 Minor Code: 253 (0X00FD)

Description
**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.DOSMUXSEMWAIT

**Minor Code**

253 (0X00FD)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Timeout=%D

OS2KRNL Major Code: 0X0005 Minor Code: 254 (0X00FE)

**Description**

(OS) DosNewSize Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSNEWSIZE

**Minor Code**

254 (0X00FE)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Filesize %W%W Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 255 (0X00FF)

**Description**

(OS) DosOpen2 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN2

**Minor Code**

255 (0X00FF)

**Trace Groups**

FS

**Trace Types**

PRE
OS2KRNL Major Code: 0X0005 Minor Code: 256 (0X0100)

Description  
(OS) DosOpen Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN

Minor Code  
256 (0X0100)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
Filename = %S  
Mode = %W Control = %W  
Attrib = %W Size = %D

OS2KRNL Major Code: 0X0005 Minor Code: 257 (0X0101)

Description  
(OS) DosOpenSem Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPENSEM

Minor Code  
257 (0X0101)

Trace Groups  
SEM

Trace Types  
PRE

Traced Parameters  
Name = %S
OS2KRNL Major Code: 0X0005 Minor Code: 258 (0X0102)

Description
(OS) DosOpLockRelease Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPLOCKRELEASE

Minor Code
258 (0X0102)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Kernel key = %W:%W Flags = %W

OS2KRNL Major Code: 0X0005 Minor Code: 259 (0X0103)

Description
(OS) DosOpLockWait Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPLOCKWAIT

Minor Code
259 (0X0103)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 260 (0X0104)

Description
(OS) DosPeekNmPipe Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPEEKNMPIPE

Minor Code
260 (0X0104)
Trace Groups: PIP
Trace Types: PRE
Traced Parameters:

Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 261 (0X0105)

Description: (OS) DosPhysicalDisk Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPHYSICALDISK
Minor Code: 261 (0X0105)
Trace Groups: FS
Trace Types: PRE
Traced Parameters:

Function = %W Data Len = %W
Parm Len = %W

OS2KRNL Major Code: 0X0005 Minor Code: 262 (0X0106)

Description: (OS) DosQNmpPipeState Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMPIPESEMSTATE
Minor Code: 262 (0X0106)
Trace Groups: PIP
Trace Types: PRE
Traced Parameters:

Semaphore handle = %W%W
OS2KRNL Major Code: 0X0005 Minor Code: 263 (0X0107)

Description
(OS) DosPortAccess Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPORTACCESS

Minor Code
263 (0X0107)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
Last Port = %W First Port = %W

OS2KRNL Major Code: 0X0005 Minor Code: 265 (0X0109)

Description
(OS) DosQCurDir Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQCURDIR

Minor Code
265 (0X0109)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Drive Number = %W

OS2KRNL Major Code: 0X0005 Minor Code: 266 (0X010A)

Description
(OS) DosQCurDisk Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQCURDISK
Minor Code: 266 (0X010A)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 267 (0X010B)

Description: (OS) DosQFHandState Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFHANDSTATE
Minor Code: 267 (0X010B)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: File Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 268 (0X010C)

Description: (OS) DosQFileInfo Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFILEINFO
Minor Code: 268 (0X010C)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: File Handle = %W Info Level = %W
OS2KRNL Major Code: 0X0005 Minor Code: 269 (0X010D)

Description
(OS) DosQFileMode Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFILEMODE

Minor Code
269 (0X010D)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
File = %S

OS2KRNL Major Code: 0X0005 Minor Code: 270 (0X010E)

Description
(OS) DosQFSAttach Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFSATTACH

Minor Code
270 (0X010E)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Path = %S
Info Level = %W

OS2KRNL Major Code: 0X0005 Minor Code: 271 (0X010F)

Description
(OS) DosQFSInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFSINFO
OS2KRNL Major Code: 0X0005 Minor Code: 271 (0X010F)

**Description**
(OS) DosQHandType Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQHANDTYPE

**Minor Code**
271 (0X010F)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Drive = %W  Info Level = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 272 (0X0110)

**Description**
(OS) DosQHandType Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQHANDTYPE

**Minor Code**
272 (0X0110)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Handle = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 273 (0X0111)

**Description**
(OS) DosQNMPHandState Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMMPHANDSTATE

**Minor Code**
273 (0X0111)

**Trace Groups**
PIP

**Trace Types**
PRE

**Traced Parameters**
Handle = %W
OS2KRNL Major Code: 0X0005 Minor Code: 274 (0X0112)

Description: (OS) DosQNmPipeInfo Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMPIPEINFO

Minor Code: 274 (0X0112)

Trace Groups: PIP

Trace Types: PRE

Traced Parameters:
- Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 275 (0X0113)

Description: (OS) DosQPathInfo Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQPATHINFO

Minor Code: 275 (0X0113)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:
- Path = %S
- Info Level = %W

OS2KRNL Major Code: 0X0005 Minor Code: 276 (0X0114)

Description: (OS) DosQSysInfo Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQSYSINFO

Minor Code
276 (0X0114)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Index = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 277 (0X0115)

Description
(OS) DosQVerify Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQVERIFY

Minor Code
277 (0X0115)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
No parameters traced.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 278 (0X0116)

Description
(OS) DosRawReadNmPipe Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRAWREADNMPipe

Minor Code
278 (0X0116)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters
OS2KRNL Major Code: 0X0005 Minor Code: 279 (0X0117)

Description
(OS) DosRawWriteNmPipe Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRAWWRITENMPIPE

Minor Code
279 (0X0117)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 280 (0X0118)

Description
(OS) DosIRead Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIREAD

Minor Code
280 (0X0118)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

File Handle = %W Buffer = %W:%W

Buffer Size = %W Pointer to bytes read = %W:%W

OS2KRNL Major Code: 0X0005 Minor Code: 282 (0X011A)

Description
(OS) DosReallocHuge Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSREALLOCHUGE

**Minor Code**
282 (0X011A)

**Trace Groups**
SEL

**Trace Types**
PRE

**Traced Parameters**

New size = %W
Selector = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 283 (0X011B)

**Description**
(OS) DosReallocSeg Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSREALLOCSEG

**Minor Code**
283 (0X011B)

**Trace Groups**
SEL

**Trace Types**
PRE

**Traced Parameters**

Selector = %W
New size = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 284 (0X011C)

**Description**
(OS) DosResumeThread Dos Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRESUMETHREAD

**Minor Code**
284 (0X011C)

**Trace Groups**
TK

**Trace Types**
PRE
Traced Parameters

Function = %W

OS2KRNL Major Code: 0X0005 Minor Code: 285 (0X011D)

Description
(OS) DosRmDir Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRMDIR

Minor Code
285 (0X011D)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

Directory = %S

OS2KRNL Major Code: 0X0005 Minor Code: 286 (0X011E)

Description
(OS) DosSelectDisk Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSELECTDISK

Minor Code
286 (0X011E)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

Drive Number = %W

OS2KRNL Major Code: 0X0005 Minor Code: 290 (0X0122)
Description: (OS) DosSemSetWait Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSEMSETWAIT

Minor Code: 290 (0X0122)

Trace Groups: SEM

Trace Types: PRE

Traced Parameters:

Handle = %W %W

OS2KRNL Major Code: 0X0005 Minor Code: 292 (0X0124)

Description: (OS) DosSendSignal Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSEND SIGNAL

Minor Code: 292 (0X0124)

Trace Groups: SIG

Trace Types: PRE

Traced Parameters:

Signal number = %W CSID = %W

OS2KRNL Major Code: 0X0005 Minor Code: 294 (0X0126)

Description: (OS) DosSetDateTime Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETDATETIME

Minor Code: 294 (0X0126)

Trace Groups: TIM

Trace Types: PRE
OS2KRNL Major Code: 0X0005 Minor Code: 295 (0X0127)

Description
(OS) DosSetFHandState Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFHANDSTATE

Minor Code
295 (0X0127)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
File Handle = %W State = %W

OS2KRNL Major Code: 0X0005 Minor Code: 296 (0X0128)

Description
(OS) DosSetFileInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFILEINFO

Minor Code
296 (0X0128)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Info Level = %W

OS2KRNL Major Code: 0X0005 Minor Code: 297 (0X0129)
**Description**  
(OS) DosSetFileMode Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFILEMODE

**Minor Code**  
297 (0X0129)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**

File = %S

---------------------------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 298 (0X012A)**

**Description**  
(OS) DosSetFSInfo Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFSINFO

**Minor Code**  
298 (0X012A)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**

Level = %W

---------------------------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 299 (0X012B)**

**Description**  
(OS) DosSetMaxFH Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETMAXFH

**Minor Code**  
299 (0X012B)

**Trace Groups**  
FS

**Trace Types**  
PRE
Traced Parameters

Handles = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 300 (0X012C)

Description
(OS) DosSetNmPHandState Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETNMHANDSTATE

Minor Code
300 (0X012C)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters

Handle = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 301 (0X012D)

Description
(OS) DosSetNmPipeSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETNMPipeSEM

Minor Code
301 (0X012D)

Trace Groups
PIP

Trace Types
PRE

Traced Parameters

Semaphore handle = %W Pipe handle = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 302 (0X012E)
Description: (OS) DosSetPathInfo Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETPATHINFO

Minor Code: 302 (0X012E)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:

Path = %S
Info Level = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 303 (0X012F)

Description: (OS) DosSetPrty Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETPRTY

Minor Code: 303 (0X012F)

Trace Groups: TK

Trace Types: PRE

Traced Parameters:

Scope = %W Priority Class, Delta = %W
ID = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 304 (0X0130)

Description: (OS) DosSetSigHandler Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETSIGHANDLER

Minor Code: 304 (0X0130)

Trace Groups
SIG

**Trace Types**
PRE

**Traced Parameters**

Signal number = %W Action = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 305 (0X0131)**

**Description**
(OS) DosSetVec Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETVEC

**Minor Code**
305 (0X0131)

**Trace Groups**
SIG

**Trace Types**
PRE

**Traced Parameters**

Return address = %W%W Vector = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 306 (0X0132)**

**Description**
(OS) DosSetVerify Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETVERIFY

**Minor Code**
306 (0X0132)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
No parameters traced.

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 307 (0X0133)**
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSleep Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSLEEP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>307 (0X0133)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>TK</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Timeout Interval = %D</td>
</tr>
</tbody>
</table>

_OS2KRNL Major Code: 0X0005 Minor Code: 308 (0X0134)_

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSuspendThread Dos Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSUSPENDTHREAD</td>
</tr>
<tr>
<td>Minor Code</td>
<td>308 (0X0134)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>TK</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Function = %W</td>
</tr>
</tbody>
</table>

_OS2KRNL Major Code: 0X0005 Minor Code: 309 (0X0135)_

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSystemService Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSYSTEMSERVICE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>309 (0X0135)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>TK</td>
</tr>
</tbody>
</table>
Trace Types
PRE

Traced Parameters
Function = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 310 (0X0136)

Description
(OS) DosTimerAsync Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IASYNCTIMER

Minor Code
310 (0X0136)

Trace Groups
TIM

Trace Types
PRE

Traced Parameters
Interval = %D
Semaphore at %D

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 311 (0X0137)

Description
(OS) DosTimerStart Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ISTARTTIMER

Minor Code
311 (0X0137)

Trace Groups
TIM

Trace Types
PRE

Traced Parameters
Interval = %D
Semaphore at %D
OS2KRNL Major Code: 0X0005 Minor Code: 312 (0X0138)

**Description**
(OS) Dos32StopTimer Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32STOPTIMER

**Minor Code**
312 (0X0138)

**Trace Groups**
TIM

**Trace Types**
PRE

**Traced Parameters**
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 313 (0X0139)

**Description**
(OS) DosTransactNmPipe Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSTRANSACTIONMPIPE

**Minor Code**
313 (0X0139)

**Trace Groups**
PIP

**Trace Types**
PRE

**Traced Parameters**
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 314 (0X013A)

**Description**
(OS) DosWaitNmPipe Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSWAITNMPIPE
Minor Code: 314 (0X013A)

Trace Groups: PIP

Trace Types: PRE

Traced Parameters:

Name = %S

OS2KRNL Major Code: 0X0005 Minor Code: 315 (0X013B)

Description: (OS) DosIWrite Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIWRITE

Minor Code: 315 (0X013B)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:

File Handle = %W Buffer = %W:%W
Buffer Size = %W Pointer to bytes written = %W:%W

OS2KRNL Major Code: 0X0005 Minor Code: 323 (0X0143)

Description: (OS) DosGetResource2 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETRESOURCE2

Minor Code: 323 (0X0143)

Trace Groups: LDR

Trace Types: PRE

Traced Parameters:
OS2KRNL Major Code: 0X0005 Minor Code: 324 (0X0144)

**Description**
(OS) DosGetResource2 Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETRESOURCE2

**Minor Code**
324 (0X0144)

**Trace Groups**
LDR

**Trace Types**
POST

**Traced Parameters**
Address = %W:%W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 325 (0X0145)

**Description**
(OS) DosFreeResource Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREERESOURCE

**Minor Code**
325 (0X0145)

**Trace Groups**
LDR

**Trace Types**
PRE

**Traced Parameters**
Selector = %W

OS2KRNL Major Code: 0X0005 Minor Code: 326 (0X0146)

**Description**
(OS) DosFreeResource Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREERESOURCE

**Minor Code**
326 (0X0146)

**Trace Groups**
LDR

**Trace Types**
POST

**Traced Parameters**
Return code = %W

------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 327 (0X0147)

**Description**
(OS) Dos32GetResource Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETRESOURCE

**Minor Code**
327 (0X0147)

**Trace Groups**
LDR

**Trace Types**
PRE

**Traced Parameters**

hMod = %D  TypeID = %D
NameID = %D

------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 328 (0X0148)

**Description**
(OS) Dos32GetResource Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETRESOURCE

**Minor Code**
328 (0X0148)

**Trace Groups**
LDR

**Trace Types**
POST

Traced Parameters

Address = %F  Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 331 (0X014B)

Description
(OS) Dos32FreeResource Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32FREERESOURCE

Minor Code
331 (0X014B)

Trace Groups
LDR

Trace Types
PRE

Traced Parameters

Pointer = %F

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 332 (0X014C)

Description
(OS) Dos32FreeResource Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32FREERESOURCE

Minor Code
332 (0X014C)

Trace Groups
LDR

Trace Types
POST

Traced Parameters

Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 333 (0X014D)
Description: (OS) Dos32QueryProcAddr Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYPROCADDR

Minor Code: 333 (0X014D)

Trace Groups: LDR

Trace Types: PRE

Traced Parameters:

Name = %S
hMod = %D Ord = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 334 (0X014E)

Description: (OS) Dos32QueryProcAddr Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYPROCADDR

Minor Code: 334 (0X014E)

Trace Groups: LDR

Trace Types: POST

Traced Parameters:

Proc Addr = %F Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 335 (0X014F)

Description: (OS) Dos32CreateEventSem Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEEVENTSEM

Minor Code: 335 (0X014F)

Trace Groups: LDR

Trace Types: PRE
SEM

Trace Types
PRE

Traced Parameters
Name ptr = %F Attribs = %D
Initial State = %D

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 336 (0X150)

Description
(OS) Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEEVENTSEM

Minor Code
336 (0X150)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Handle = %D Return code = %D

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 337 (0X151)

Description
(OS) Dos32OpenEventSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENEVENTSEM

Minor Code
337 (0X151)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Name Ptr = %F phev = %F

---------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 338 (0X0152)

Description: (OS) Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENEVENTSEM

Minor Code: 338 (0X0152)

Trace Groups: SEM

Trace Types: POST

Traced Parameters:

Handle = %D Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 339 (0X0153)

Description: (OS) Dos32OpenEventSem Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEEVENTSEM

Minor Code: 339 (0X0153)

Trace Groups: SEM

Trace Types: PRE

Traced Parameters:

Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 340 (0X0154)

Description: (OS) Dos32CloseEventSem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEEVENTSEM

Minor Code: 
OS2KRNL Major Code: 0X0005 Minor Code: 341 (0X0155)

Description
(OS) Dos32ResetEventSem Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32RESETEVENTSEM
Minor Code
341 (0X0155)
Trace Groups
SEM
Trace Types
PRE
Traced Parameters
Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 342 (0X0156)

Description
(OS) Dos32ResetEventSem Post-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32RESETEVENTSEM
Minor Code
342 (0X0156)
Trace Groups
SEM
Trace Types
POST
Traced Parameters
Return code = %D
OS2KRNL Major Code: 0X0005 Minor Code: 343 (0X0157)

**Description**  
(OS) Dos32PostEventSem Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32POSTEVENTSEM

**Minor Code**  
343 (0X0157)

**Trace Groups**  
SEM

**Trace Types**  
PRE

**Traced Parameters**

Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 344 (0X0158)

**Description**  
(OS) Dos32PostEventSem Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32POSTEVENTSEM

**Minor Code**  
344 (0X0158)

**Trace Groups**  
SEM

**Trace Types**  
POST

**Traced Parameters**

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 345 (0X0159)

**Description**  
(OS) Dos32WaitEvenSem Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITEVENTSEM
Minor Code: 345 (0X0159)
Trace Groups: SEM
Trace Types: PRE
Traced Parameters:
Handle = %D Timeout = %D

OS2KRNL Major Code: 0X0005 Minor Code: 346 (0X015A)

Description: (OS) Dos32WaitEventSem Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITEVENTSEM
Minor Code: 346 (0X015A)
Trace Groups: SEM
Trace Types: POST
Traced Parameters:
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 347 (0X015B)

Description: (OS) Dos32QueryEventSem Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYEVENTSEM
Minor Code: 347 (0X015B)
Trace Groups: SEM
Trace Types: PRE
Traced Parameters:
Handle = %D
OS2KRNL Major Code: 0X0005 Minor Code: 348 (0X015C)

Description
(OS) Dos32QueryEventSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYEVENTSEM

Minor Code
348 (0X015C)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
State = %D Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 349 (0X015D)

Description
(OS) Dos32CreateMutexSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEMUTEXSEM

Minor Code
349 (0X015D)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Name ptr = %F Attribs = %D
Initial State = %D

OS2KRNL Major Code: 0X0005 Minor Code: 350 (0X015E)

Description
(OS) Dos32CreateMutexSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEMUTEXSEM

**Minor Code**
350 (0X015E)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**
Handle = %D Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 351 (0X015F)

**Description**
(OS) Dos32OpenMutexSem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENMUTEXSEM

**Minor Code**
351 (0X015F)

**Trace Groups**
SEM

**Trace Types**
PRE

**Traced Parameters**
Name ptr = %F Handle = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 352 (0X0160)

**Description**
(OS) Dos32OpenMutexSem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENMUTEXSEM

**Minor Code**
352 (0X0160)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**
OS2KRNL Major Code: 0X0005 Minor Code: 353 (0X0161)

Description
(OS) Dos32CloseMutexSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEMUTEXSEM

Minor Code
353 (0X0161)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 354 (0X0162)

Description
(OS) Dos32CloseMutexSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEMUTEXSEM

Minor Code
354 (0X0162)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 355 (0X0163)

Description
(OS) Dos32RequestMutexSem Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32REQUESTMUTEXSEM

Minor Code
355 (0X0163)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Handle = %D Timeout = %D

OS2KRNL Major Code: 0X0005 Minor Code: 356 (0X0164)

Description
(OS) Dos32RequestMutexSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32REQUESTMUTEXSEM

Minor Code
356 (0X0164)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 357 (0X0165)

Description
(OS) Dos32ReleaseMutexSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32RELEASEMUTEXSEM

Minor Code
357 (0X0165)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
OS2KRNL Major Code: 0X0005 Minor Code: 358 (0X0166)

Description
(OS) Dos32ReleaseMutexSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32RELEASEMUTEXSEM

Minor Code
358 (0X0166)

Trace Groups
SEM

Trace Types
POST

Traced Parameters

Return code = %D

-------------------------------------------------
**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMUTEXSEM

**Minor Code**
360 (0X0168)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**

PID Owner = %D TID Owner = %D
Count = %D Return code = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 361 (0X0169)**

**Description**
(OS) Dos32CreateMuxWaitSem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEMUXWAITSEM

**Minor Code**
361 (0X0169)

**Trace Groups**
SEM

**Trace Types**
PRE

**Traced Parameters**

Name ptr = %F cSemRec = %D
SemRec ptr = %F Attribs = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 362 (0X016A)**

**Description**
(OS) Dos32CreateMuxWaitSem Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEMUXWAITSEM

**Minor Code**
362 (0X016A)

**Trace Groups**
SEM
Trace Types

POST

Traced Parameters

Handle = %D Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 363 (0X016B)

Description

(OS) Dos32OpenMuxWaitSem Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENMUXWAITSEM

Minor Code

363 (0X016B)

Trace Groups

SEM

Trace Types

PRE

Traced Parameters

Name Ptr = %F phmux = %F

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 364 (0X016C)

Description

(OS) Dos32OpenMuxWaitSem Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENMUXWAITSEM

Minor Code

364 (0X016C)

Trace Groups

SEM

Trace Types

POST

Traced Parameters

Handle = %D Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 365 (0X016D)
(OS) Dos32CloseMuxWaitSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEMUXWAITSEM

Minor Code
365 (0X016D)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 366 (0X016E)

Description
(OS) Dos32CloseMuxWaitSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEMUXWAITSEM

Minor Code
366 (0X016E)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 367 (0X016F)

Description
(OS) Dos32WaitMuxWaitSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITMUXWAITSEM

Minor Code
367 (0X016F)

Trace Groups
SEM
Trace Types
PRE

Traced Parameters
Handle = %D Timeout = %D

OS2KRNL Major Code: 0X0005 Minor Code: 368 (0X0170)

Description
(OS) Dos32WaitMuxWaitSem Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITMUXWAITSEM

Minor Code
368 (0X0170)

Trace Groups
SEM

Trace Types
POST

Traced Parameters
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 369 (0X0171)

Description
(OS) Dos32AddMuxWaitSem Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ADDMUXWAITSEM

Minor Code
369 (0X0171)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Handle = %D SemRec = %Q

OS2KRNL Major Code: 0X0005 Minor Code: 370 (0X0172)
Description: (OS) Dos32AddMuxWaitSem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ADDMUXWAITSEM

Minor Code: 370 (0X0172)

Trace Groups: SEM

Trace Types: POST

Traced Parameters:

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 371 (0X0173)

Description: (OS) Dos32DeleteMuxWaitSem Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DELETEMUXWAITSEM

Minor Code: 371 (0X0173)

Trace Groups: SEM

Trace Types: PRE

Traced Parameters:

Mux Handle = %D Sem Handle = %D

OS2KRNL Major Code: 0X0005 Minor Code: 372 (0X0174)

Description: (OS) Dos32DeleteMuxWaitSem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DELETEMUXWAITSEM

Minor Code: 372 (0X0174)

Trace Groups: SEM
**Trace Types**

POST

**Traced Parameters**

Return code = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 373 (0X0175)**

**Description**

(OS) Dos32QueryMuxWaitSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMUXWAITSEM

**Minor Code**

373 (0X0175)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Handle = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 374 (0X0176)**

**Description**

(OS) Dos32QueryMuxWaitSem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMUXWAITSEM

**Minor Code**

374 (0X0176)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

\[cSemRec = %D\]
\[pSemRec = %F\]
\[Attribs = %D\]
\[Return code = %D\]
OS2KRNL Major Code: 0X0005 Minor Code: 375 (0X0177)

**Description**  
(OS) DosGetCP Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETCP

**Minor Code**  
375 (0X0177)

**Trace Groups**  
NLS

**Trace Types**  
PRE

**Traced Parameters**  
No parameters traced.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 376 (0X0178)

**Description**  
(OS) DosGetCP Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETCP

**Minor Code**  
376 (0X0178)

**Trace Groups**  
NLS

**Trace Types**  
POST

**Traced Parameters**  
No parameters traced.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 377 (0X0179)

**Description**  
(OS) Dos32AsyncTimer Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ASYNCTIMER

**Minor Code**  
377 (0X0179)

**Trace Groups**  
TIM
Trace Types
PRE

Traced Parameters
Interval = %D
Semaphore at %D

OS2KRNL Major Code: 0X0005 Minor Code: 378 (0X017A)

Description
(OS) Dos32AsyncTimer Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ASYNCTIMER

Minor Code
378 (0X017A)

Trace Groups
TIM

Trace Types
POST

Traced Parameters
Return code = %D Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 379 (0X017B)

Description
(OS) Dos32StartTimer Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32STARTTIMER

Minor Code
379 (0X017B)

Trace Groups
TIM

Trace Types
PRE

Traced Parameters
Interval = %D
Semaphore at %D
OS2KRNL Major Code: 0X0005 Minor Code: 380 (0X017C)

**Description**  
(OS) Dos32StartTimer Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32STARTTIMER

**Minor Code**  
380 (0X017C)

**Trace Groups**  
TIM

**Trace Types**  
POST

**Traced Parameters**

```
Return code = %D Handle = %W
```

-----------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 391 (0X0187)

**Description**  
(OS) Dos32WaitThread Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITTHREAD

**Minor Code**  
391 (0X0187)

**Trace Groups**  
SEM

**Trace Types**  
PRE

**Traced Parameters**

```
ThreadID = %D WaitOption = %D
```

-----------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 392 (0X0188)

**Description**  
(OS) Dos32WaitThread Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITTHREAD

**Minor Code**
Trace Groups
SEM

Trace Types
POST

Traced Parameters

ThreadID = %D Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 393 (0X0189)

Description
(OS) Cluster Allocate Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.Allocate

Minor Code
393 (0X0189)

Trace Groups
FS

Trace Types
PRE

Traced Parameters

Cluster Count Wanted = %W

OS2KRNL Major Code: 0X0005 Minor Code: 394 (0X018A)

Description
(OS) Cluster Allocate Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postAllocate

Minor Code
394 (0X018A)

Trace Groups
FS

Trace Types
POST

Traced Parameters

Cluster Count Obtained = %W
OS2KRNL Major Code: 0X0005 Minor Code: 395 (0X018B)

**Description**
(OS) Cluster Release Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preRelease

**Minor Code**
395 (0X018B)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Cluster Release = %W

OS2KRNL Major Code: 0X0005 Minor Code: 397 (0X018D)

**Description**
(OS) File Lock/Unlock Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.w_LockOper

**Minor Code**
397 (0X018D)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Unlock Flag = %W Handle = %W
Region Offset = %D Length = %D

OS2KRNL Major Code: 0X0005 Minor Code: 401 (0X0191)

**Description**
(OS) Thread Reschedule Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postSchedNext

**Minor Code**
401 (0X0191)

**Trace Groups**
TK

**Trace Types**
POST

**Traced Parameters**
Switched to PID = %W TID = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 403 (0X0193)

**Description**
(OS) DosMuxSemWait Post-Invocation (No Wait)

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMUXSEMWAIT2

**Minor Code**
403 (0X0193)

**Trace Groups**
SEM

**Trace Types**
PRE

**Traced Parameters**
Index = %W Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 404 (0X0194)

**Description**
(OS) DosEnumAttribute Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSENUMATTRIBUTE

**Minor Code**
404 (0X0194)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
No parameters traced.
OS2KRNL Major Code: 0X0005 Minor Code: 405 (0X0195)

Description  
(OS) DosEnumAttribute Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSENUMATTRIBUTE

Minor Code  
405 (0X0195)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 406 (0X0196)

Description  
(OS) DosISetFileInfo Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETFILEINFO

Minor Code  
406 (0X0196)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  

Info Level = %W

OS2KRNL Major Code: 0X0005 Minor Code: 407 (0X0197)

Description  
(OS) DosISetFileInfo Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETFILEINFO
OS2KRNRL Major Code: 0X0005 Minor Code: 407 (0X0197)

Description
(OS) DosISetPathInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNRL.preDOSISETPATHINFO

OS2KRNRL Major Code: 0X0005 Minor Code: 408 (0X0198)

Description
(OS) DosISetPathInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNRL.preDOSISETPATHINFO

Minor Code
408 (0X0198)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Path = %S
Info Level = %W

OS2KRNRL Major Code: 0X0005 Minor Code: 409 (0X0199)

Description
(OS) DosISetPathInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNRL.postDOSISETPATHINFO

Minor Code
409 (0X0199)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 410 (0X019A)

Description
(OS) Dos32QueryResource Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYRESOURCESIZE

Minor Code
410 (0X019A)

Trace Groups
LDR

Trace Types
PRE

Traced Parameters

hMod = %D TypeID = %D
NameID = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 411 (0X019B)

Description
(OS) Dos32QueryResourceSize Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYRESOURCESIZE

Minor Code
411 (0X019B)

Trace Groups
LDR

Trace Types
POST

Traced Parameters

Address = %F Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 412 (0X019C)

Description
(OS) DosIDevIoctl Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIDEVIOCTL

**Minor Code**
412 (0X019C)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
No parameters traced.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 413 (0X019D)

**Description**
(OS) DosIDevIoctl Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIDEVIOCTL

**Minor Code**
413 (0X019D)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 414 (0X019E)

**Description**
(OS) DosISetRelMaxFH Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETRELMAXFH

**Minor Code**
414 (0X019E)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
No parameters traced.
OS2KRNL Major Code: 0X0005 Minor Code: 415 (0X019F)

**Description**
(OS) DosISetRelMaxFH Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETRELMAXFH

**Minor Code**
415 (0X019F)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 416 (0X01A0)

**Description**
(OS) Dos32InitializePorthole Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32INITIALIZEPORTHOLE

**Minor Code**
416 (0X01A0)

**Trace Groups**
LDR

**Trace Types**
PRE

**Traced Parameters**
Init Routine = %D entry type = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 417 (0X01A1)

**Description**
(OS) Dos32InitializePorthole Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32INITIALIZEPORTHOLE

**Minor Code**: 417 (0X01A1)

**Trace Groups**: LDR

**Trace Types**: POST

**Traced Parameters**

Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 418 (0X01A2)

**Description**

(OS) Dos32QueryHeaderInfo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYHEADERINFO

**Minor Code**: 418 (0X01A2)

**Trace Groups**: LDR

**Trace Types**: PRE

**Traced Parameters**

hMod = %D index = %D
phdr = %D cb = %D
subfunc = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 419 (0X01A3)

**Description**

(OS) Dos32QueryHeaderInfo Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYHEADERINFO

**Minor Code**: 419 (0X01A3)

**Trace Groups**: LDR

**Trace Types**: POST
OS2KRNL Major Code: 0X0005 Minor Code: 420 (0X01A4)

**Description**  
(OS) Dos32QueryProcType Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYPROCtypeName

**Minor Code**  
420 (0X01A4)

**Trace Groups**  
LDR

**Trace Types**  
PRE

**Traced Parameters**

Name = %S  
hMod = %D Ord = %D

OS2KRNL Major Code: 0X0005 Minor Code: 421 (0X01A5)

**Description**  
(OS) Dos32QueryProcType Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYPROCtypeName

**Minor Code**  
421 (0X01A5)

**Trace Groups**  
LDR

**Trace Types**  
POST

**Traced Parameters**

Proc Type = %F Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 424 (0X01A8)
**Description**
(OS) DosOpen2Compt Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN2COMPT

**Minor Code**
424 (0X01A8)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**

Filename = %S
Mode = %W Control = %W
Attrib = %W Size = %D

--------------------------------------------

**Description**
(OS) DosOpen2Compt Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN2COMPT

**Minor Code**
425 (0X01A9)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**

Action = %W Handle = %W
Return Code = %W

--------------------------------------------

**Description**
(OS) Dos32ISetFHState Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ISETFHSTATE

**OS2KRNL Major Code: 0X0005 Minor Code: 425 (0X01A9)**

**OS2KRNL Major Code: 0X0005 Minor Code: 428 (0X01AC)**
Minor Code: 428 (0X01AC)
Trace Groups: FS
Trace Types: PRE
Traced Parameters:
File Handle = %W  State = %W %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 429 (0X01AD)

Description: (OS) Dos32ISetFHState Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ISETFHSTATE
Minor Code: 429 (0X01AD)
Trace Groups: FS
Trace Types: POST
Traced Parameters:
Return Code = %W

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 430 (0X01AE)

Description: (OS) Dos32IQUERYFHSTATE Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IQUERYFHSTATE
Minor Code: 430 (0X01AE)
Trace Groups: FS
Trace Types: PRE
Traced Parameters:
File Handle = %W
OS2KRNL Major Code: 0X0005 Minor Code: 431 (0X01AF)

Description
(OS) Dos32IQUERYFHSTATE Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IQUERYFHSTATE

Minor Code
431 (0X01AF)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Handle State = %W Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 432 (0X01B0)

Description
(OS) Dos32IRead Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IREAD

Minor Code
432 (0X01B0)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
File Handle = %D Buffer = %F
Buffer Size = %D Pointer to Bytes Read = %F

OS2KRNL Major Code: 0X0005 Minor Code: 433 (0X01B1)

Description
(OS) Dos32IRead Post-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IREAD

Minor Code
433 (0X01B1)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Bytes Read = %D Return code = %D

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 434 (0X01B2)

Description
(OS) Dos32IWrite Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IWRITE

Minor Code
434 (0X01B2)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
File Handle = %D Buffer = %F
Buffer Size = %D Pointer to Bytes Written = %F

---------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 435 (0X01B3)

Description
(OS) Dos32IWrite Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IWRITE

Minor Code
435 (0X01B3)

Trace Groups
FS

Trace Types
POST
OS2KRNL Major Code: 0X0005 Minor Code: 436 (0X01B4)

Description
(OS) Dos32DumpProcess Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DUMPPROCESS

Minor Code
436 (0X01B4)

Trace Groups
TK

Trace Types
PRE

Traced Parameters
Flag = %D Drive = %D
PID = %D

OS2KRNL Major Code: 0X0005 Minor Code: 437 (0X01B5)

Description
(OS) Dos32DumpProcess Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DUMPPROCESS

Minor Code
437 (0X01B5)

Trace Groups
TK

Trace Types
POST

Traced Parameters
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 438 (0X01B6)
Description: (OS) Dos32SuppressPopUps Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SUPPRESSPOPUPS

Minor Code: 438 (0X01B6)

Trace Groups: TK

Trace Types: PRE

Traced Parameters:
Flag = %D Drive = %D

OS2KRNL Major Code: 0X0005 Minor Code: 439 (0X01B7)

Description: (OS) Dos32SuppressPopups Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SUPPRESSPOPUPS

Minor Code: 439 (0X01B7)

Trace Groups: TK

Trace Types: POST

Traced Parameters:
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 440 (0X01B8)

Description: (OS) Dos32KillThread Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32KILLTHREAD

Minor Code: 440 (0X01B8)

Trace Groups: TK
Trace Types: PRE

Traced Parameters:

Tid = %D

OS2KRNL Major Code: 0X0005 Minor Code: 441 (0X01B9)

Description: (OS) Dos32KillThread Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32KILLTHREAD

Minor Code: 441 (0X01B9)

Trace Groups: TK

Trace Types: POST

Traced Parameters:

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 442 (0X01BA)

Description: (OS) Dos32IProtectSetFHState Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IPROTECTSETFHSTATE

Minor Code: 442 (0X01BA)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:

File Handle = %W State = %W %W

OS2KRNL Major Code: 0X0005 Minor Code: 443 (0X01BB)
**Description**
(OS) Dos32IProtectSetFHState Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRN.L.postDOS32IPROTECTSETFHSTATE

**Minor Code**
443 (0X01BB)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Return Code = %W

-----------------------------

**OS2KRN.L Major Code: 0X0005 Minor Code: 444 (0X01BC)**

**Description**
(OS) Dos32IprotectqueryFHState Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRN.L.preDOS32IprotectqueryFHSTATE

**Minor Code**
444 (0X01BC)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
File Handle = %W

-----------------------------

**OS2KRN.L Major Code: 0X0005 Minor Code: 445 (0X01BD)**

**Description**
(OS) Dos32IprotectqueryFHState Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRN.L.postDOS32IprotectqueryFHSTATE

**Minor Code**
445 (0X01BD)

**Trace Groups**
FS
Trace Types

POST

Traced Parameters

Handle State = %W Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 446 (0X01BE)

Description

(OS) DosProtectChgFilePtr Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTCHGFILEPTR

Minor Code

446 (0X01BE)

Trace Groups

FS

Trace Types

PRE

Traced Parameters

Type = %W Distance = %W%W
Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 447 (0X01BF)

Description

(OS) DosProtectChgFilePtr Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTCHGFILEPTR

Minor Code

447 (0X01BF)

Trace Groups

FS

Trace Types

POST

Traced Parameters

Location = %W%W Return code = %W

--------------------------------------------
OS2KRNL Major Code: 0X0005 Minor Code: 448 (0X01C0)

**Description**
(OS) DosProtectClose Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTCLOSE

**Minor Code**
448 (0X01C0)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**

Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 449 (0X01C1)

**Description**
(OS) DosProtectClose Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTCLOSE

**Minor Code**
449 (0X01C1)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 450 (0X01C2)

**Description**
(OS) DosCloseChangeNotify Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSECHANGENOTIFY

**Minor Code**
450 (0X01C2)
OS2KRNL Major Code: 0X0005 Minor Code: 451 (0X01C3)

**Description**
(OS) DosCloseChangeNotify Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSECHANGENOTIFY

**Minor Code**
451 (0X01C3)

**Traced Parameters**
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 452 (0X01C4)

**Description**
(OS) DosProtectEnumAttribute Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTENUMATTRIBUTE

**Minor Code**
452 (0X01C4)

**Traced Parameters**
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 453 (0X01C5)
(OS) DosProtectEnumAttribute Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTENUMATTRIBUTE

Minor Code
453 (0X01C5)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 454 (0X01C6)

(OS) DosProtectFileIO Pre-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTFILEIO

Minor Code
454 (0X01C6)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 455 (0X01C7)

(OS) DosProtectFileIO Post-Invocation

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTFILEIO

Minor Code
455 (0X01C7)

Trace Groups
FS
Trace Types

POST

Traced Parameters

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 456 (0X01C8)

Description

(OS) DosProtectFileLocks Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTFILELOCKS

Minor Code

456 (0X01C8)

Trace Groups

FS

Trace Types

PRE

Traced Parameters

Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 457 (0X01C9)

Description

(OS) DosProtectFileLocks Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTFILELOCKS

Minor Code

457 (0X01C9)

Trace Groups

FS

Trace Types

POST

Traced Parameters

Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 458 (0X01CA)
Description: (OS) DosForceDelete Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFORCEDELETE

Minor Code: 458 (0X01CA)

Trace Groups: FS

Trace Types: PRE

Traced Parameters:
Path = %S

OS2KRNL Major Code: 0X0005 Minor Code: 459 (0X01CB)

Description: (OS) DosForceDelete Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFORCEDELETE

Minor Code: 459 (0X01CB)

Trace Groups: FS

Trace Types: POST

Traced Parameters:
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 460 (0X01CC)

Description: (OS) DosIProtectRead Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIPROTECTREAD

Minor Code: 460 (0X01CC)

Trace Groups: FS
**Trace Types**
PRE

**Traced Parameters**
Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 461 (0X01CD)

**Description**
(OS) DosIProtectRead Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSI PROTECTREAD

**Minor Code**
461 (0X01CD)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**
Bytes Read = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 462 (0X01CE)

**Description**
(OS) DosIProtectSetFileInfo Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSI PROTECTSETFILEINFO

**Minor Code**
462 (0X01CE)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Info Level = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 463 (0X01CF)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosIProtectSetFileInfo Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIPROTECTSETFILEINFO</td>
</tr>
<tr>
<td>Minor Code</td>
<td>463 (0X01CF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosIProtectWrite Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIPROTECTWRITE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>464 (0X01D0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Handle = %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosIProtectWrite Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIPROTECTWRITE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>465 (0X01D1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
</tbody>
</table>
Trace Types
POST

Traced Parameters
Bytes Written = %W Return code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 466 (0X01D2)

Description
(OS) DosProtectNewSize Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTNEWSIZE

Minor Code
466 (0X01D2)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Filesize %W%W Handle = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 467 (0X01D3)

Description
(OS) DosProtectNewSize Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTNEWSIZE

Minor Code
467 (0X01D3)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return Code = %W

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 468 (0X01D4)
**Description**  
(OS) DOSPROTECTOPEN Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTOPEN

**Minor Code**  
468 (0X01D4)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**

- Filename = %S
- Mode = %W Control = %W
- Attrib = %W Size = %D

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 469 (0X01D5)**

**Description**  
(OS) DosProtectOpen Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTOPEN

**Minor Code**  
469 (0X01D5)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**

- Action = %W Handle = %W
- Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 470 (0X01D6)**

**Description**  
(OS) DosOpenChangeNotify Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPENCHANGENOTIFY
Minor Code: 470 (0X01D6)  
Trace Groups: FS  
Trace Types: PRE  
Traced Parameters: No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 471 (0X01D7)

Description: (OS) DosOpenChangeNotify Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPENCHANGENOTIFY
Minor Code: 471 (0X01D7)  
Trace Groups: FS  
Trace Types: POST  
Traced Parameters:  
Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 472 (0X01D8)

Description: (OS) DosProtectQHandState Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTQFHANDSTATE
Minor Code: 472 (0X01D8)  
Trace Groups: FS  
Trace Types: PRE  
Traced Parameters:  
File Handle = %W
OS2KRNL Major Code: 0X0005 Minor Code: 473 (0X01D9)

Description
(OS) DosProtectQfHandState Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTQFHANDSTATE

Minor Code
473 (0X01D9)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Handle State = %W Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 474 (0X01DA)

Description
(OS) DosProtectQFileInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTQFILEINFO

Minor Code
474 (0X01DA)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
File Handle = %W Info Level = %W

OS2KRNL Major Code: 0X0005 Minor Code: 475 (0X01DB)

Description
(OS) DosProtectQFileInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTQFILEINFO
OS2KRNL Major Code: 0X0005 Minor Code: 476 (0X01DC)

Description
(OS) DosResetChangeNotify Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRESETCHANGENOTIFY

Minor Code
476 (0X01DC)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 477 (0X01DD)

Description
(OS) DosResetChangeNotify Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRESETCHANGENOTIFY

Minor Code
477 (0X01DD)

Trace Groups
FS

Trace Types
POST

Traced Parameters
Return code = %W
<table>
<thead>
<tr>
<th>OS2KRNL Major Code: 0X0005 Minor Code: 478 (0X01DE)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OS2KRNL Major Code: 0X0005 Minor Code: 479 (0X01DF)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OS2KRNL Major Code: 0X0005 Minor Code: 480 (0X01E0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
</tbody>
</table>
480 (0X01E0)

**Trace Groups**
FS

**Trace Types**
PRE

**Traced Parameters**
Info Level = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 481 (0X01E1)**

**Description**
(OS) DosProtectSetFileInfo Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTSETFILEINFO

**Minor Code**
481 (0X01E1)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 482 (0X01E2)**

**Description**
(OS) Dos32PMPostEvenSem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMPOSTEVENTSEM

**Minor Code**
482 (0X01E2)

**Trace Groups**
SEM

**Trace Types**
PRE

**Traced Parameters**

PMHandle = %D Handle = %D
OS2KRNL Major Code: 0X0005 Minor Code: 483 (0X01E3)

**Description**  
(OS) Dos32PMPostEventSem Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMPOSTEVENTSEM

**Minor Code**  
483 (0X01E3)

**Trace Groups**  
SEM

**Trace Types**  
POST

**Traced Parameters**

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 484 (0X01E4)

**Description**  
(OS) Dos32PMWaitEventSem Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMWEAITEVENTSEM

**Minor Code**  
484 (0X01E4)

**Trace Groups**  
SEM

**Trace Types**  
POST

**Traced Parameters**

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 485 (0X01E5)

**Description**  
(OS) Dos32PMWaitMuxWaitSem Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMWAITMUXWAITSEM
Minor Code: 485 (0X01E5)

Trace Groups: SEM

Trace Types: PRE

Traced Parameters:

Handle = %D
Handle = %D
Timeout = %D

OS2KRNL Major Code: 0X0005 Minor Code: 486 (0X01E6)

Description: (OS) Dos32PMWaitMuxWaitSem Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMWAITMUXWAITSEM

Minor Code: 486 (0X01E6)

Trace Groups: SEM

Trace Types: POST

Traced Parameters:

Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 487 (0X01E7)

Description: (OS) Dos32QueryExLibPATH Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYEXTLIBPATH

Minor Code: 487 (0X01E7)

Trace Groups: VM

Trace Types: PRE

Traced Parameters:
OS2KRNL Major Code: 0X0005 Minor Code: 488 (0X01E8)

- **Description**: (OS) Dos32QueryExtLIBPATH Post-Invocation
- **Tracepoint**: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYEXTLIBPATH
- **Minor Code**: 488 (0X01E8)
- **Trace Groups**: VM
- **Trace Types**: POST
- **Traced Parameters**:
  - Path = %S
  - Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 489 (0X01E9)

- **Description**: (OS) Dos32SetExtLIBPATH Pre-Invocation
- **Tracepoint**: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETEXTLIBPATH
- **Minor Code**: 489 (0X01E9)
- **Trace Groups**: VM
- **Trace Types**: PRE
- **Traced Parameters**:
  - Path = %S
  - Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 490 (0X01EA)

- **Description**: (OS) Dos32SetExtLIBPATH Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETEXLIBPATH

Minor Code: 490 (0X01EA)

Trace Groups: VM

Trace Types: POST

Traced Parameters:

Return code = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 491 (0X01EB)

Description: (OS) Dos32VERIFYPIDTID Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32VERIFYPIDTID

Minor Code: 491 (0X01EB)

Trace Groups: TK

Trace Types: PRE

Traced Parameters:

Pid = %D Tid = %D

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 492 (0X01EC)

Description: (OS) Dos32VERIFYPIDTID Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32VERIFYPIDTID

Minor Code: 492 (0X01EC)

Trace Groups: TK

Trace Types: POST

Traced Parameters:
Return code = %D

OS2KRNL Major Code: 0X0005 Minor Code: 493 (0X01ED)

**Description**
(OS) Dos32PMWaitEvenSem Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMWAITEVENTSEM

**Minor Code**
493 (0X01ED)

**Trace Groups**
SEM

**Trace Types**
PRE

**Traced Parameters**
PMHandle = %D Handle = %D
Timeout = %D

OS2KRNL Major Code: 0X0005 Minor Code: 0494 (0X01EE)

**Description**
(OS) Dos32CancelLockRequest Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CANCELLOCKREQUEST

**Minor Code**
494 (0X01EE)

**Trace Groups**
FS

**Trace Types**
PRE, API

**Traced Parameters**
Offset = %F Range = %F Handle = %D

*Note:* Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

OS2KRNL Major Code: 0X0005 Minor Code: 00495 (0X01EF)
Description: (OS) Dos32CancelLockRequest Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CANCELLOCKREQUEST

Minor Code: 495 (0X01EF)

Trace Groups: FS

Trace Types: PRE, API

Traced Parameters:

Return code = %D

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00496 (0X01F0)

Description: (OS) Dos32SetFileLocks Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETFILELOCKS

Minor Code: 496 (0X01F0)

Trace Groups: FS

Trace Types: PRE, API

Traced Parameters:

Handle = D%
Unlock Offset = %F Range = %F*
Lock Offset = %F Range = %F*
Timeout = %D Flags = %D*

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00497 (0X01F1)
### OS2KRNL Major Code: 0X0005 Minor Code: 00497 (0X01F1)

**Description**  
(OS) Dos32SetFileLocks Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETFILELOCKS

**Minor Code**  
497 (0X01F1)

**Trace Groups**  
FS

**Trace Types**  
PRE, API

**Traced Parameters**

Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

### OS2KRNL Major Code: 0X0005 Minor Code: 00498 (0X01F2)

**Description**  
(OS) Dos32ProtectSetFileLocks Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PROTECTSETFILELOCKS

**Minor Code**  
498 (0X01F2)

**Trace Groups**  
FS

**Trace Types**  
PRE, API

**Traced Parameters**

Handle = %D

Unlock Offset = %F Range = %F

Lock Offset = %F Range = %F

Timeout = %D Flags = %D Lock Id = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

### OS2KRNL Major Code: 0X0005 Minor Code: 00499 (0X01F3)

**Description**  
(OS) Dos32ProtectSetFileLocks Post-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PROTECTSETFILELOCKS

Minor Code
499 (0X01F3)

Trace Groups
FS

Trace Types
PRE, API

Traced Parameters

Return code = %D

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00500 (0X01F4)

Description
(OS) DosCreateSpinLock Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATESPINLOCK

Minor Code
500 (0X01F4)

Trace Groups
LOCK

Trace Types
PRE, API

Traced Parameters

pLockHandle = %W:%W

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00501 (0X01F5)

Description
(OS) DosCreateSpinLock Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATESPINLOCK

Minor Code
501 (0X01F5)

Trace Groups
LOCK

Trace Types
POST, API

Traced Parameters

Return code = %D

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00502 (0X01F6)

Description
(OS) DosAcquireSpinLock Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSACQUIRESPINLOCK

Minor Code
502 (0X01F6)

Trace Groups
LOCK

Trace Types
PRE, API

Traced Parameters

"Lock Handle = %W%W

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00503 (0X01F7)

Description
(OS) DosAcquireSpinLock Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSACQUIRESPINLOCK

Minor Code
503 (0X01F7)

Trace Groups
LOCK

Trace Types
POST, API

Traced Parameters
No parameters traced.
Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

OS2KRNL Major Code: 0X0005 Minor Code: 00504 (0X01F8)

Description  (OS) DosReleaseSpinLock Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRELEASESPINLOCK
Minor Code  504 (0X01F8)
Trace Groups  LOCK
Trace Types  PRE, API
Traced Parameters  "Lock Handle = %W%W"

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

OS2KRNL Major Code: 0X0005 Minor Code: 00505 (0X01F9)

Description  (OS) DosReleaseSpinLock Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRELEASESPINLOCK
Minor Code  505 (0X01F8)
Trace Groups  LOCK
Trace Types  POST, API
Traced Parameters  No parameters traced.

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

OS2KRNL Major Code: 0X0005 Minor Code: 00506 (0X01FA)
Description: (OS) DosFreeSpinLock Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREESPINLOCK

Minor Code: 506 (0X01FA)

Trace Groups: LOCK

Trace Types: PRE, API

Traced Parameters:

"Lock Handle = %W%W

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00507 (0X01FB)

Description: (OS) DosFreeSpinLock Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREESPINLOCK

Minor Code: 5071 (0X01FB)

Trace Groups: LOCK

Trace Types: POST, API

Traced Parameters:

Return code = %D

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35.

--------------------------------------------

OS2KRNL Major Code: 0X0005 Minor Code: 00508 (0X01FC)

Description: (OS) Dos32IProtectRead Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PROTECTSETFILELOCKS

---
**Minor Code**
508 (0X01FC)

**Trace Groups**
FS

**Trace Types**
PRE, API

**Traced Parameters**

File Handle = %D Buffer = %F
Buffer Size = %D Pointer to Bytes Read = %F

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 00509 (0X01FD)**

**Description**
(OS) Dos32IProtectRead Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTREAD

**Minor Code**
509 (0X01FD)

**Trace Groups**
FS

**Trace Types**
POST, API

**Traced Parameters**

Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

**OS2KRNL Major Code: 0X0005 Minor Code: 00510 (0X01FE)**

**Description**
(OS) Dos32IProtectWrite Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IPROTECTWRITE

**Minor Code**
510 (0X01FE)

**Trace Groups**
FS
Trace Types

PRE, API

Traced Parameters

File Handle = %D Buffer = %F
Buffer Size = %D Pointer to Bytes Written = %F

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

OS2KRNL Major Code: 0X0005 Minor Code: 00511 (0X01FF)

Description

(OS) Dos32IProtectWrite Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTWRITE

Minor Code

511 (0X01FF)

Trace Groups

FS

Trace Types

POST, API

Traced Parameters

Return code = %D

Note: Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

OS2KRNL Major Code: 0X0005 Minor Code: 32768 (0X8000)

Description

(OS) DosSetTraceInfo Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETTRACEINFO

Minor Code

32768 (0X8000)

Trace Groups

TK

Trace Types

PRE

Traced Parameters
OS2KRNL Major Code: 0X0005 Minor Code: 65521 (0XFFF1)

Description
(OS) DosSetTraceInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETTRACEINFO

Minor Code
65521 (0XFFF1)

Trace Groups
TK

Trace Types
POST

Traced Parameters

Return code = %W

-------------------------------

DevHlp Services Trace Events

The tracepoints for the Device Helper Services major code are identified in the following tables. These tracepoints are static tracepoints. They are compiled with the code.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for DEVHELP Major Code: 0X0006, sorted by minor code.
Trace events for DEVHELP Major Code: 0X0006 ,sorted by tracepoint.

-------------------------------

Trace Events for DEVHELP Major Code: 0X0006, Sorted by Minor Code

00001 (0X0001) (OS) DevHlp_ABIOSCall Pre-Invocation
00002 (0X0002) (OS) DevHlp_ABIOSCommonEnt Pre-Invocation
00003 (0X0003) (OS) DevHlp_ABIOSGetParms Pre-Invocation
00004 (0X0004) (OS) DevHlp_AddTraceEvent Pre-Invocation
00005 (0X0005) (OS) DevHlp_AllocGDTSel Pre-Invocation
00006 (0X0006) (OS) DevHlp_AllocPhys Pre-Invocation
00007 (0X0007) (OS) DevHlp_AllocReqPacket Pre-Invocation
00008 (0X0008) (OS) DevHlp_AttachDD Pre-Invocation
00009 (0X0009) (OS) DevHlp_Block Pre-Invocation
00010 (0X000A) (OS) DevHlp_DeRegister Pre-I invocation
00011 (0X000B) (OS) DevHlp_DeDone Pre-I invocation
00012 (0X000C) (OS) DevHlp_EOL Pre-I invocation
00013 (0X000D) (OS) DevHlp_FreeLIDEntry Pre-I invocation
00014 (0X000E) (OS) DevHlp_FreePhys Pre-I invocation
00015 (0X000F) (OS) DevHlp_FreeReqPacket Pre-I invocation
00016 (0X0010) (OS) DevHlp_GetDeviceBlock Pre-I invocation
00017 (0X0011) (OS) DevHlp_GetDOSVar Pre-I invocation
00018 (0X0012) (OS) DevHlp_GetLIDEntry Pre-I invocation
00019 (0X0013) (OS) DevHlp_InternalError Pre-I invocation
00020 (0X0014) (OS) DevHlp_Lock Pre-I invocation
00021 (0X0015) (OS) DevHlp_LogEntry Pre-I invocation
00022 (0X0016) (OS) DevHlp_MonCreate Pre-I invocation
00023 (0X0017) (OS) DevHlp_MonFlush Pre-I invocation
00024 (0X0018) (OS) DevHlp_MonWrite Pre-I invocation
00025 (0X0019) (OS) DevHlp_PhysToGDTSelector Pre-I invocation
00026 (0X001A) (OS) DevHlp_PhysToUVirt Pre-I invocation
00027 (0X001B) (OS) DevHlp_PhysToVirt Pre-I invocation
00028 (0X001C) (OS) DevHlp_Profiling Kernel Pre-I invocation
00029 (0X001D) (OS) DevHlp_ProtToReal Pre-I invocation
00030 (0X001E) (OS) DevHlp_PullParticular Pre-I invocation
00031 (0X001F) (OS) DevHlp_PullReqPacket Pre-I invocation
00032 (0X0020) (OS) DevHlp_PushReqPacket Pre-I invocation
00033 (0X0021) (OS) DevHlp_QueueFlush Pre-I invocation
00034 (0X0022) (OS) DevHlp_QueueInit Pre-I invocation
00035 (0X0023) (OS) DevHlp_QueueRead Pre-I invocation
00036 (0X0024) (OS) DevHlp_QueueWrite Pre-I invocation
00037 (0X0025) (OS) DevHlp_RealToProt Pre-I invocation
00038 (0X0026) (OS) DevHlp_Register Pre-I invocation
00039 (0X0027) (OS) DevHlp_RegStackUsage Pre-I invocation
00040 (0X0028) (OS) DevHlp_ResetTimer Pre-I invocation
00041 (0X0029) (OS) DevHlp_ROMCritSection Pre-I invocation
00042 (0X002A) (OS) DevHlp_RompCritSection Pre-I invocation
00043 (0X002B) (OS) DevHlp_ProcRun Pre-I invocation
00044 (0X002C) (OS) DevHlp_SchedClock Pre-I invocation
00045 (0X002D) (OS) DevHlp_SemClear Pre-I invocation
00046 (0X002E) (OS) DevHlp_SemHandle Pre-I invocation
00047 (0X002F) (OS) DevHlp_SemRequest Pre-I invocation
00048 (0X0030) (OS) DevHlp_SendEvent Pre-I invocation
00049 (0X0031) (OS) DevHlp_SetIRQ Pre-I invocation
00050 (0X0032) (OS) DevHlp_SetROMVector Pre-I invocation
00051 (0X0033) (OS) DevHlp_SetTimer Pre-I invocation
00052 (0X0034) (OS) DevHlp_SortReqPacket Pre-I invocation
00053 (0X0035) (OS) DevHlp_TCYield Pre-I invocation
00054 (0X0036) (OS) DevHlp_TickCount Pre-I invocation
00055 (0X0037) (OS) DevHlp.Unlock Pre-I invocation
00056 (0X0038) (OS) DevHlp.UnPhysToVirt Pre-I invocation
00057 (0X0039) (OS) DevHlp.UnSetIRQ Pre-I invocation
00058 (0X003A) (OS) DevHlp.VerifyAccess Pre-I invocation
00059 (0X003B) (OS) DevHlp.VirtualToPhys Pre-I invocation
00060 (0X003C) (OS) DevHlp.VMAllocate Pre-I invocation
00061 (0X003D) (OS) DevHlp.VMGlobalToProcess Pre-I invocation
00062 (0X003E) (OS) DevHlp.VMGlobalToBlock Pre-I invocation
00063 (0X003F) (OS) DevHlp.VMUnlock Pre-I invocation
00064 (0X0040) (OS) DevHlp.VMUnlock Pre-I invocation
00065 (0X0041) (OS) DevHlp.VMUnlock Pre-I invocation
00066 (0X0042) (OS) DevHlp.VMProcessToGlobal Pre-I invocation
00067 (0X0043) (OS) DevHlp.VMUnlock Pre-I invocation
00068 (0X0044) (OS) DevHlp.Yield Pre-I invocation
00069 (0X0045) (OS) DevHlp.Yield Pre-I invocation
00070 (0X0046) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00071 (0X0047) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00072 (0X0048) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00073 (0X0049) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00074 (0X004A) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00075 (0X004B) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00076 (0X004C) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00077 (0X004D) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00078 (0X004E) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00079 (0X004F) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00080 (0X0050) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00081 (0X0051) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00082 (0X0052) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00083 (0X0053) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00084 (0X0054) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00085 (0X0055) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00086 (0X0056) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00087 (0X0057) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00088 (0X0058) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00089 (0X0059) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00090 (0X005A) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00091 (0X005B) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00092 (0X005C) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00093 (0X005D) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00094 (0X005E) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00095 (0X005F) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00096 (0X0060) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00097 (0X0061) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00098 (0X0062) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00099 (0X0063) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00100 (0X0064) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00101 (0X0065) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00102 (0X0066) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00103 (0X0067) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00104 (0X0068) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00105 (0X0069) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00106 (0X006A) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00107 (0X006B) (OS) DevHlp.FreeGDTSelector Pre-I invocation
00108 (0X006C) (OS) DevHlp.ModifyPriority Pre-I invocation
00109 (0X006D) (OS) DevHlp_RegisterTmrDD Pre-Invocation
00110 (0X006E) (OS) DevHlp_RegisterPerfCtrs Pre-Invocation
00124 (0X007C) (OS) DevHlp_RegisterPDD Pre-Invocation
00125 (0X007D) (OS) DevHlp_RegisterBeep Pre-Invocation
00126 (0X007E) (OS) DevHlp_Beep Pre-Invocation
00129 (0X0081) (OS) DevHlp_ABIOSCall Post-Invocation
00130 (0X0082) (OS) DevHlp_ABIOSCommonEnt Post-Invocation
00131 (0X0083) (OS) DevHlp_ABIOSGetParms Post-Invocation
00132 (0X0084) (OS) DevHlp_AddTraceEvent Post-Invocation
00133 (0X0085) (OS) DevHlp_AllocGDTSel Post-Invocation
00134 (0X0086) (OS) DevHlp_AlocPhys Post-Invocation
00135 (0X0087) (OS) DevHlp_AllocReqPacket Post-Invocation
00136 (0X0088) (OS) DevHlp_AttachDD Post-Invocation
00137 (0X0089) (OS) DevHlp_Block Post-Invocation
00138 (0X008A) (OS) DevHlp_DeRegister Post-Invocation
00139 (0X008B) (OS) DevHlp_DevDone Post-Invocation
00140 (0X008C) (OS) DevHlp_EOI Post-Invocation
00141 (0X008D) (OS) DevHlp_FreeLIDEntry Post-Invocation
00142 (0X008E) (OS) DevHlp_FreePhys Post-Invocation
00143 (0X008F) (OS) DevHlp_FreeReqPacket Post-Invocation
00144 (0X0090) (OS) DevHlp_GetDeviceBlock Post-Invocation
00145 (0X0091) (OS) DevHlp_GetDOSVar Post-Invocation
00146 (0X0092) (OS) DevHlp_GetLIDEntry Post-Invocation
00147 (0X0093) (OS) DevHlp_InternalError Post-Invocation
00148 (0X0094) (OS) DevHlp_Lock Post-Invocation
00149 (0X0095) (OS) DevHlp_LogEntry Post-Invocation
00150 (0X0096) (OS) DevHlp_MonCreate Post-Invocation
00151 (0X0097) (OS) DevHlp_MonFlush Post-Invocation
00152 (0X0098) (OS) DevHlp_MonWrite Post-Invocation
00153 (0X0099) (OS) DevHlp_PhysToGDTSelector Post-Invocation
00154 (0X009A) (OS) DevHlp_PhysToUVirt Post-Invocation
00155 (0X009B) (OS) DevHlp_ProfileKernel Post-Invocation
00156 (0X009C) (OS) DevHlp_ProfileProcess Post-Invocation
00157 (0X009D) (OS) DevHlp_ProfToReal Post-Invocation
00158 (0X009E) (OS) DevHlp_PullParticular Post-Invocation
00159 (0X009F) (OS) DevHlp_PullReqPacket Post-Invocation
00160 (0X00A0) (OS) DevHlp_PushReqPacket Post-Invocation
00161 (0X00A1) (OS) DevHlp_Callback Post-Invocation
00162 (0X00A2) (OS) DevHlp_Flush Post-Invocation
00163 (0X00A3) (OS) DevHlp_QueueRead Post-Invocation
00164 (0X00A4) (OS) DevHlp_PullReqPacket Post-Invocation
00165 (0X00A5) (OS) DevHlp_RealToProt Post-Invocation
00166 (0X00A6) (OS) DevHlp_Register Post-Invocation
00167 (0X00A7) (OS) DevHlp_RegStackUsage Post-Invocation
00168 (0X00A8) (OS) DevHlp_ResetTimer Post-Invocation
00170 (0X00A9) (OS) DevHlp_ROMCritSection Post-Invocation
00171 (0X00AA) (OS) DevHlp_ProcRun Post-Invocation
00172 (0X00AB) (OS) DevHlp_SchedClock Post-Invocation
00173 (0X00AC) (OS) DevHlp_SemClear Post-Invocation
00174 (0X00AE) (OS) DevHlp_SemHandle Post-Invocation
00175 (0X00AF) (OS) DevHlp_SemRequest Post-Invocation
00176 (0X00B0) (OS) DevHlp_SendEvent Post-Invocation
00177 (0X00B1) (OS) DevHlp_SetIRQ Post-Invocation
00178 (0X00B2) (OS) DevHlp_SetROMVector Post-Invocation
00179 (0X00B3) (OS) DevHlp_SetTimer Post-Invocation
00180 (0X00B4) (OS) DevHlp_SortReqPacket Post-Invocation
00181 (0X00B5) (OS) DevHlp_TickCount Post-Invocation
00182 (0X00B6) (OS) DevHlp_TickCount Post-Invocation
00183 (0X00B7) (OS) DevHlp_TickCount Post-Invocation
00184 (0X00B8) (OS) DevHlp_UnPhysToVirt Post-Invocation
00185 (0X00B9) (OS) DevHlp_UnSetIRQ Post-Invocation
00186 (0X00BA) (OS) DevHlp_VerifyAccess Post-Invocation
00188 (0X00BC) (OS) DevHlp_VirtToPhys Post-Invocation
00189 (0X00BD) (OS) DevHlp_VMAlloc Post-Invocation
00190 (0X00BE) (OS) DevHlp_VMFree Post-Invocation
00191 (0X00BF) (OS) DevHlp_VMGlobalToProcess Post-Invocation
00193 (0X00C1) (OS) DevHlp_VMLock Post-Invocation
00194 (0X00C2) (OS) DevHlp_VMProcessToGlobal Post-Invocation
00195 (0X00C3) (OS) DevHlp_VMUnlock Post-Invocation
00197 (0X00C5) (OS) DevHlp_Yield Post-Invocation
00201 (0X00C9) (OS) DevHlp_FreeGDTSelector Post-Invocation
00202 (0X00CA) (OS) DevHlp_VirtToLin Post-Invocation
00203 (0X00CB) (OS) DevHlp_LinToGDTSelector Post-Invocation
Trace Events for DEVHELP Major Code: 0X0006, Sorted by Tracepoint

(OS) DevHlp_ABIOSCall Post-Invocation 00129 (0X0081)
(OS) DevHlp_ABIOSCall Pre-Invocation 00001 (0X0001)
(OS) DevHlp_ABIOSCommonEnt Post-Invocation 00130 (0X0082)
(OS) DevHlp_ABIOSCommonEnt Pre-Invocation 00002 (0X0002)
(OS) DevHlp_ABIOSGetParms Post-Invocation 00131 (0X0083)
(OS) DevHlp_ABIOSGetParms Pre-Invocation 00003 (0X0003)
(OS) DevHlp_AddTraceEvent Post-Invocation 00132 (0X0084)
(OS) DevHlp_AddTraceEvent Pre-Invocation 00004 (0X0004)
(OS) DevHlp_AllocCtxHook Post-Invocation 00211 (0X00D3)
(OS) DevHlp_AllocCtxHook Pre-Invocation 00083 (0X0053)
(OS) DevHlp_AllocGDTSel Post-Invocation 00133 (0X0085)
(OS) DevHlp_AllocGDTSel Pre-Invocation 00005 (0X0005)
(OS) DevHlp_AllocPhys Post-Invocation 00134 (0X0086)
(OS) DevHlp_AllocPhys Pre-Invocation 00006 (0X0006)
(OS) DevHlp_AllocReqPacket Post-Invocation 00135 (0X0087)
(OS) DevHlp_AllocReqPacket Pre-Invocation 00007 (0X0007)
(OS) DevHlp_ArmCtxHook Post-Invocation 00212 (0X00D4)
(OS) DevHlp_ArmCtxHook Pre-Invocation 00084 (0X0054)
(OS) DevHlp_AttachDD Post-Invocation 00136 (0X0088)
(OS) DevHlp_AttachDD Pre-Invocation 00008 (0X0008)
(OS) DevHlp_Beep Post-Invocation 00254 (0X00FE)
(OS) DevHlp_Beep Pre-Invocation 00126 (0X007E)
(OS) DevHlp_Block Post-Invocation 00137 (0X0089)
(OS) DevHlp_Block Pre-Invocation 00009 (0X0009)
(OS) DevHlp_CloseEventSem Post-Invocation 00216 (0X00D8)
(OS) DevHlp_CloseEventSem Pre-Invocation 00088 (0X0058)
(OS) DevHlp_DeRegister Post-Invocation 00138 (0X008A)
(OS) DevHlp_DeRegister Pre-Invocation 00010 (0X000A)
(OS) DevHlp_DevDone Post-Invocation 00139 (0X008B)
(OS) DevHlp_DevDone Pre-Invocation 00011 (0X000B)
(OS) DevHlp_EOI Post-Invocation 00140 (0X008C)
(OS) DevHlp_EOI Pre-Invocation 00012 (0X000C)
(OS) DevHlp_FreeCtxHook Post-Invocation 00213 (0X00D5)
(OS) DevHlp_FreeCtxHook Pre-Invocation 00085 (0X0055)
(OS) DevHlp_FreeGDTSelSelector Post-Invocation 00201 (0X00C9)
(OS) DevHlp_FreeGDTSelSelector Pre-Invocation 00073 (0X0049)
(OS) DevHlp_FreeLIDEntry Post-Invocation 00141 (0X008D)
(OS) DevHlp_FreeLIDEntry Pre-Invocation 00013 (0X000D)
(OS) DevHlp_FreePhys Post-Invocation 00142 (0X008E)
(OS) DevHlp_FreePhys Pre-Invocation 00014 (0X000E)
(OS) DevHlp_FreeReqPacket Post-Invocation 00143 (0X008F)
(OS) DevHlp_FreeReqPacket Pre-Invocation 00015 (0X000F)
(OS) DevHlp_GetDOSVar Post-Invocation 00145 (0X0091)
(OS) DevHlp_GetDOSVar Pre-Invocation 00017 (0X0011)
(OS) DevHlp_GetDescInfo Post-Invocation 00204 (0X00CC)
(OS) DevHlp_GetDescInfo Pre-Invocation 00076 (0X004C)
(OS) DevHlp_GetDeviceBlock Post-Invocation 00144 (0X0090)
(OS) DevHlp_GetDeviceBlock Pre-Invocation 00016 (0X0010)
(OS) DevHlp_GetLIDEntry Post-Invocation 00146 (0X0092)
(OS) DevHlp_GetLIDEntry Pre-Invocation 00018 (0X0012)
(OS) DevHlp_InternalError Post-Invocation 00147 (0X0093)
(OS) DevHlp_InternalError Pre-Invocation 00019 (0X0013)
(OS) DevHlp_LinToGDTSelector Post-Invocation 00203 (0X00CB)
(OS) DevHlp_LinToGDTSelector Pre-Invocation 00075 (0X004B)
(OS) DevHlp_LinToPageList Post-Invocation 00205 (0X00CD)
(OS) DevHlp_LinToPageList Pre-Invocation 00077 (0X004D)
(OS) DevHlp_Lock Post-Invocation 00148 (0X0094)
(OS) DevHlp_Lock Pre-Invocation 00020 (0X0014)
(OS) DevHlp_LogEntry Post-Invocation 00149 (0X0095)
(OS) DevHlp_LogEntry Pre-Invocation 00021 (0X0015)
(OS) DevHlp_ModifyPriority Post-Invocation 00236 (0X00EC)
(OS) DevHlp_ModifyPriority Pre-Invocation 00108 (0X006C)
(OS) DevHlp_MonCreate Post-Invocation 00150 (0X0096)
(OS) DevHlp_MonCreate Pre-Invocation 00022 (0X0016)
(OS) DevHlp_MonFlush Post-Invocation 00151 (0X0097)
(OS) DevHlp_MonFlush Pre-Invocation 00023 (0X0017)
(OS) DevHlp_MonWrite Post-Invocation 00152 (0X0098)
(OS) DevHlp_MonWrite Pre-Invocation 00024 (0X0018)
(OS) DevHlp_OpenEventSem Post-Invocation 00215 (0X00D7)
(OS) DevHlp_OpenEventSem Pre-Invocation 00087 (0X0057)
(OS) DevHlp_PageListToGDTSelector Post-Invocation 00207 (0X00CF)
(OS) DevHlp_PageListToGDTSelector Pre-Invocation 00079 (0X004F)
(OS) DevHlp_PageListToLin Post-Invocation 00206 (0X00CE)
(OS) DevHlp_PageListToLin Pre-Invocation 00078 (0X004E)
(OS) DevHlp_PhysToGDTSet Post-Invocation 00209 (0X00D1)
(OS) DevHlp_PhysToGDTSet Pre-Invocation 00081 (0X0051)
(OS) DevHlp_PhysToGDTSelector Post-Invocation 00153 (0X0099)
(OS) DevHlp_PhysToGDTSelector Pre-Invocation 00025 (0X0019)
(OS) DevHlp_PhysToUVirt Post-Invocation 00154 (0X009A)
(OS) DevHlp_PhysToUVirt Pre-Invocation 00026 (0X001A)
(OS) DevHlp_PhysToVirt Post-Invocation 00155 (0X009B)
(OS) DevHlp_PhysToVirt Pre-Invocation 00027 (0X001B)
(OS) DevHlp_PostEventSem Post-Invocation 00217 (0X00D9)
(OS) DevHlp_PostEventSem Pre-Invocation 00089 (0X0059)
(OS) DevHlp_ProcRun Post-Invocation 00171 (0X00AB)
(OS) DevHlp_ProcRun Pre-Invocation 00043 (0X002B)
(OS) DevHlp_ProcRun2 Post-Invocation 00221 (0X00DD)
(OS) DevHlp_ProcRun2 Pre-Invocation 00093 (0X005D)
(OS) DevHlp_Profiling Kernel Post-Invocation 00156 (0X009C)
(OS) DevHlp_Profiling Kernel Pre-Invocation 00028 (0X001C)
(OS) DevHlp_ProtToReal Post-Invocation 00157 (0X009D)
(OS) DevHlp_ProtToReal Pre-Invocation 00029 (0X001D)
(OS) DevHlp_PullParticular Post-Invocation 00158 (0X009E)
(OS) DevHlp_PullParticular Pre-Invocation 00030 (0X001E)
(OS) DevHlp_PullReqPacket Post-Invocation 00159 (0X009F)
(OS) DevHlp_PullReqPacket Pre-Invocation 00031 (0X001F)
(OS) DevHlp_PushReqPacket Post-Invocation 00160 (0X00A0)
(OS) DevHlp_PushReqPacket Pre-Invocation 00032 (0X0020)
(OS) DevHlp_QueueFlush Post-Invocation 00161 (0X00A1)
(OS) DevHlp_QueueFlush Pre-Invocation 00033 (0X0021)
(OS) DevHlp_QueueInit Post-Invocation 00162 (0X00A2)
(OS) DevHlp_QueueInit Pre-Invocation 00034 (0X0022)
(OS) DevHlp_QueueRead Post-Invocation 00163 (0X00A3)
(OS) DevHlp_QueueRead Pre-Invocation 00035 (0X0023)
(OS) DevHlp_QueueWrite Post-Invocation 00164 (0X00A4)
(OS) DevHlp_QueueWrite Pre-Invocation 00036 (0X0024)
(OS) DevHlp_ROMCritSection Post-Invocation 00170 (0X00AA)
(OS) DevHlp_ROMCritSection Pre-Invocation 00042 (0X002A)
(OS) DevHlp_RegStackUsage Post-Invocation 00167 (0X00A7)
(OS) DevHlp_RegStackUsage Pre-Invocation 00039 (0X0027)
(OS) DevHlp_Register Post-Invocation 00166 (0X00A6)
(OS) DevHlp_Register Pre-Invocation 00038 (0X0026)
Tracepoint
Static tracepoint in DEVHELP.

Minor Code
1 (0X0001)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
LiD=%W Entry_Point=%W ABIOS Request Block=%A

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 2 (0X0002)

Description
(OS) DevHlp_ABIOSCommonEnt Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
2 (0X0002)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 3 (0X0003)

Description
(OS) DevHlp_ABIOSGetParms Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
3 (0X0003)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

--------------------------------------------
DEVHELP Major Code: 0X0006 Minor Code: 4 (0X0004)

**Description**
(OS) DevHlp_AddTraceEvent Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
4 (0X0004)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 5 (0X0005)

**Description**
(OS) DevHlp_AllocGDTSel Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
5 (0X0005)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

pSelectorArray=%A   Number of GDT Selectors=%W

DEVHELP Major Code: 0X0006 Minor Code: 6 (0X0006)

**Description**
(OS) DevHlp_AllocPhys Pre-Invocation
Tracepoint Static tracepoint in DEVHELP.

Minor Code 6 (0X0006)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

size_low=%W size_high=%W relative position=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 7 (0X0007)

Description (OS) DevHlp_AllocReqPacket Pre-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 7 (0X0007)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters No parameters traced.

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 8 (0X0008)

Description (OS) DevHlp_AttachDD Pre-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 8 (0X0008)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters No parameters traced.
DEVHELP Major Code: 0X0006 Minor Code: 9 (0X0009)

**Description**  
(OS) DevHlp_Block Pre-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
9 (0X0009)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

- event_id_low=%W  
- event_id_high=%W  
- timeout_interval=%D  
- interruptible_flag=%W

DEVHELP Major Code: 0X0006 Minor Code: 10 (0X000A)

**Description**  
(OS) DevHlp_DeRegister Pre-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
10 (0X000A)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

- monitor_PID=%W  
- monitor_handle=%W

DEVHELP Major Code: 0X0006 Minor Code: 11 (0X000B)

**Description**  
(OS) DevHlp_DevDone Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
11 (0X000B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 12 (0X000C)

**Description**
(OS) DevHlp_EOI Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
12 (0X000C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
IRQNum=%W

DEVHELP Major Code: 0X0006 Minor Code: 13 (0X000D)

**Description**
(OS) DevHlp_FreeLIDEntry Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
13 (0X000D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
LID=%W Device Driver DS=%W
DEVHELP Major Code: 0X0006 Minor Code: 14 (0X000E)

**Description**
(OS) DevHlp_FreePhys Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
14 (0X000E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

address_low=%W address_high=%W

DEVHELP Major Code: 0X0006 Minor Code: 15 (0X000F)

**Description**
(OS) DevHlp_FreeReqPacket Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
15 (0X000F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

request_packet=%A

DEVHELP Major Code: 0X0006 Minor Code: 16 (0X0010)

**Description**
(OS) DevHlp_GetDeviceBlock Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_GetDOSVar Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>17 (0X0011)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>variable index=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 18 (0X0012)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_GetLIDEntry Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>18 (0X0012)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DeviceId=%W RelativeID#=%W</td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 19 (0X0013)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_InternalError Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>19 (0X0013)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 20 (0X0014)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_Lock Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>20 (0X0014)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 21 (0X0015)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_LogEntry Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 22 (0X0016)

Description
(OS) DevHlp_MonCreate Pre-Invocation
Tracepoint
Static tracepoint in DEVHELP.
Minor Code
22 (0X0016)
Trace Groups
No groups assigned.
Trace Types
No types assigned.
Traced Parameters
final_buffer=%A notify_rtn=%A Handle=%W

DEVHELP Major Code: 0X0006 Minor Code: 23 (0X0017)

Description
(OS) DevHlp_MonFlush Pre-Invocation
Tracepoint
Static tracepoint in DEVHELP.
Minor Code
23 (0X0017)
Trace Groups
No groups assigned.
Trace Types
No types assigned.
Traced Parameters
monitor_handle=%W
DEVHELP Major Code: 0X0006 Minor Code: 24 (0X0018)

**Description**  
(OS) DevHlp_MonWrite Pre-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
24 (0X0018)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

- data_record=%A
- count=%W
- monitor_handle=%W
- wait_flag=%W
- timeout_high=%W
- timeout_low=%W

DEVHELP Major Code: 0X0006 Minor Code: 25 (0X0019)

**Description**  
(OS) DevHlp_PhysToGDTSelector Pre-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
25 (0X0019)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

- Address=%D
- Size=%W
- Selector=%W

DEVHELP Major Code: 0X0006 Minor Code: 26 (0X001A)

**Description**  
(OS) DevHlp_PhysToUVirt Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
26 (0X001A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
address_low=%W address_high=%W
segment size=%W function code=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 27 (0X001B)

**Description**
(OS) DevHlp_PhysToVirt Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
27 (0X001B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
address to convert=%D

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 28 (0X001C)

**Description**
(OS) DevHlp_Profiling Kernel Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
28 (0X001C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DEVHELP Major Code: 0X0006 Minor Code: 29 (0X001D)

Description    (OS) DevHlp_ProtToReal Pre-Invocation
Tracepoint     Static tracepoint in DEVHELP.
Minor Code     29 (0X001D)
Trace Groups   No groups assigned.
Trace Types    No types assigned.
Traced Parameters
Device Header=%A

DEVHELP Major Code: 0X0006 Minor Code: 30 (0X001E)

Description    (OS) DevHlp_PullParticular Pre-Invocation
Tracepoint     Static tracepoint in DEVHELP.
Minor Code     30 (0X001E)
Trace Groups   No groups assigned.
Trace Types    No types assigned.
Traced Parameters
Queue=%A request_packet=%A

DEVHELP Major Code: 0X0006 Minor Code: 31 (0X001F)

Description    (OS) DevHlp_PullReqPacket Pre-Invocation
Tracepoint  Static tracepoint in DEVHELP.

Minor Code  31 (0X001F)

Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters

Queue=%A

---------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 32 (0X0020)

Description  (OS) DevHlp_PushReqPacket Pre-Invocation

Tracepoint  Static tracepoint in DEVHELP.

Minor Code  32 (0X0020)

Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters

Queue=%A request_packet=%A

---------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 33 (0X0021)

Description  (OS) DevHlp_QueueFlush Pre-Invocation

Tracepoint  Static tracepoint in DEVHELP.

Minor Code  33 (0X0021)

Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters
DEVHELP Major Code: 0X0006 Minor Code: 34 (0X0022)

Description
(OS) DevHip_QueueInit Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
34 (0X0022)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Queue=%A

DEVHELP Major Code: 0X0006 Minor Code: 35 (0X0023)

Description
(OS) DevHip_QueueRead Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
35 (0X0023)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Queue=%A

DEVHELP Major Code: 0X0006 Minor Code: 36 (0X0024)

Description
(OS) DevHip_QueueWrite Pre-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Static tracepoint in DEVHELP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>36 (0X0024)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Queue=%A Character written=%B</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 37 (0X0025)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_RealToProt Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>37 (0X0025)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Device Header=%A</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 38 (0X0026)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_Register Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>38 (0X0026)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 39 (0X0027)

**Description**
(OS) DevHlp_RegStackUsage Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
39 (0X0027)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 40 (0X0028)

**Description**
(OS) DevHlp_ResetTimer Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
40 (0X0028)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 42 (0X002A)

**Description**
(OS) DevHlp_ROMCritSection Pre-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Static tracepoint in DEVHELP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>42 (0X002A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>enter_exit flag=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 43 (0X002B)

| Description                                           | (OS) DevHlp_ProcRun Pre-Invocation               |
| Tracepoint                                           | Static tracepoint in DEVHELP.                     |
| Minor Code                                           | 43 (0X002B)                                      |
| Trace Groups                                         | No groups assigned.                              |
| Trace Types                                          | No types assigned.                               |
| Traced Parameters                                    | event_id_low=%W event_id_high=%W                  |

DEVHELP Major Code: 0X0006 Minor Code: 44 (0X002C)

| Description                                           | (OS) DevHlp_SchedClock Pre-Invocation             |
| Tracepoint                                           | Static tracepoint in DEVHELP.                     |
| Minor Code                                           | 44 (0X002C)                                      |
| Trace Groups                                         | No groups assigned.                              |
| Trace Types                                          | No types assigned.                               |
| Traced Parameters                                    | No parameters traced.                            |
DEVHELP Major Code: 0X0006 Minor Code: 45 (0X002D)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SemClear Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>45 (0X002D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>sem_handle_low=%W sem_handle_high=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 46 (0X002E)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SemHandle Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>46 (0X002E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>sem_key_low=%W sem_key_high=%W usage flag=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 47 (0X002F)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SemRequest Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td></td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 47 (0X002F)

**Description**
(OS) DevHlp_SendEvent Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
47 (0X002F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- sem_handle_low=%W
- sem_handle_high=%W
- timeout value=%D

-----------------------------

DEVHELP Major Code: 0X0006 Minor Code: 48 (0X0030)

**Description**
(OS) DevHlp_SendEvent Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
48 (0X0030)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- Event=%W

-----------------------------

DEVHELP Major Code: 0X0006 Minor Code: 49 (0X0031)

**Description**
(OS) DevHlp_SetIRQ Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
49 (0X0031)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SetROMVector Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>50 (0X0032)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SetTimer Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>51 (0X0033)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_SortReqPacket Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td></td>
</tr>
</tbody>
</table>

IRQnum=\%W
DEVHELP Major Code: 0X0006 Minor Code: 53 (0X0035)

Description  (OS) DevHlp_TCYield Pre-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  53 (0X0035)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 54 (0X0036)

Description  (OS) DevHlp_TickCount Pre-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  54 (0X0036)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  No parameters traced.
DEVHELP Major Code: 0X0006 Minor Code: 55 (0X0037)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_Unlock Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>55 (0X0037)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>lock_handle_low=%W lock_handle_high=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 56 (0X0038)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_UnPhysToVirt Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>56 (0X0038)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 57 (0X0039)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_UnSetIRQ Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>57 (0X0039)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 58 (0X003A)

**Description**

(OS) DevHip_VerifyAccess Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

58 (0X003A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

- Target Selector/Segment=%W Length=%W
- Offset=%W Type of Access=%W

-------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 60 (0X003C)

**Description**

(OS) DevHip_VirtToPhys Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

60 (0X003C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

- address=%D

-------------------------------
DEVHELP Major Code: 0X0006 Minor Code: 61 (0X003D)

**Description**
(OS) DevHlp_VMAlloc Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
61 (0X003D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Flags=%D Size=%D PhysAddr=%F

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 62 (0X003E)

**Description**
(OS) DevHlp_VMFree Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
62 (0X003E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Linear address=%F

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 63 (0X003F)

**Description**
(OS) DevHlp_VMGlobalToProcess Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.
DEVHELP Major Code: 0X0006 Minor Code: 65 (0X0041)

Description
(OS) DevHlp_VMLock Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
65 (0X0041)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Action flags=%D Linear address=%F Length=%D
pPageList=%F  pLockHandle=%F

DEVHELP Major Code: 0X0006 Minor Code: 66 (0X0042)

Description
(OS) DevHlp_VMProcessToGlobal Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
66 (0X0042)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Action flags=%D Linear address=%F Length=%D
DEVHELP Major Code: 0X0006 Minor Code: 67 (0X0043)

Description: (OS) DevHlp_VMUnlock Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 67 (0X0043)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
LockHandle=%d%d%d

DEVHELP Major Code: 0X0006 Minor Code: 69 (0X0045)

Description: (OS) DevHlp_Yield Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 69 (0X0045)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 73 (0X0049)

Description: (OS) DevHlp_FreeGDTSelector Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
DEVHELP Major Code: 0X0006 Minor Code: 74 (0X004A)

Description: (OS) DevHlp_VirtToLin Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 74 (0X004A)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
Selector=%W Offset=%D

DEVHELP Major Code: 0X0006 Minor Code: 75 (0X004B)

Description: (OS) DevHlp_LinToGDTSelector Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 75 (0X004B)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
Selector=%W Linear Address=%F Size=%D
DEVHELP Major Code: 0X0006 Minor Code: 76 (0X004C)

Description: (OS) DevHlp_GetDescInfo Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 76 (0X004C)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
Selector=%W

DEVHELP Major Code: 0X0006 Minor Code: 77 (0X004D)

Description: (OS) DevHlp_LinToPageList Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 77 (0X004D)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
Linear Address=%F Size=%D pPageList=%F

DEVHELP Major Code: 0X0006 Minor Code: 78 (0X004E)

Description: (OS) DevHlp_PageListToLin Pre-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 78 (0X004E)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

Size=%D pPageList=%F

------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 79 (0X004F)

Description: (OS) DevHlp_PageListToGDTSelector Pre-Invocation

Tracepoint: Static tracepoint in DEVHELP.

Minor Code: 79 (0X004F)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

Selector=%W Size=%D Access=%D pPageList=%F

------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 81 (0X0051)

Description: (OS) DevHlp_PhysToGDTSel Pre-Invocation

Tracepoint: Static tracepoint in DEVHELP.

Minor Code: 81 (0X0051)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

Address=%D Size=%D Access=%W Selector=%W
DEVHELP Major Code: 0X0006 Minor Code: 83 (0X0053)

**Description**
(OS) DevHlp_AllocCtxHook Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
83 (0X0053)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Hook Handler=%D Context=%D

DEVHELP Major Code: 0X0006 Minor Code: 84 (0X0054)

**Description**
(OS) DevHlp_ArmCtxHook Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
84 (0X0054)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Hook Data=%D Hook Handle=%D Context=%D

DEVHELP Major Code: 0X0006 Minor Code: 85 (0X0055)

**Description**
(OS) DevHlp_FreeCtxHook Pre-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.
DEVHELP Major Code: 0X0006 Minor Code: 87 (0X0057)

Description  (OS) DevHlp_OpenEventSem Pre-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  87 (0X0057)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  
    Sem Handle=%D

DEVHELP Major Code: 0X0006 Minor Code: 88 (0X0058)

Description  (OS) DevHlp_CloseEventSem Pre-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  88 (0X0058)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  
    Sem Handle=%D
DEVHELP Major Code: 0X0006 Minor Code: 89 (0X0059)

Description
(OS) DevHip_PostEventSem Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
89 (0X0059)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Sem Handle=%D

DEVHELP Major Code: 0X0006 Minor Code: 90 (0X005A)

Description
(OS) DevHip_ResetEventSem Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
90 (0X005A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Sem Handle=%D, Post Count=%D

DEVHELP Major Code: 0X0006 Minor Code: 93 (0X005D)

Description
(OS) DevHip_ProcRun2 Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.
Minor Code
93 (0X005D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 108 (0X006C)

Description
(OS) DevHlp_ModifyPriority Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
108 (0X006C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Thread number=\%W Keyboard switch=\%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 109 (0X006D)

Description
(OS) DevHlp_RegisterTmrDD Pre-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
109 (0X006D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PTTTimer0 entry point=\%A

--------------------------------------------
DEVHELP Major Code: 0X0006 Minor Code: 110 (0X006E)

Description (OS) DevHlp_RegisterPerfCtrs Pre-Invocation
Tracepoint Static tracepoint in DEVHELP.
Minor Code 110 (0X006E)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

pCounterBlock=%A pTextBlock=%A Flags=%W

DEVHELP Major Code: 0X0006 Minor Code: 124 (0X007C)

Description (OS) DevHlp_RegisterPDD Pre-Invocation
Tracepoint Static tracepoint in DEVHELP.
Minor Code 124 (0X007C)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

Entry point=%A PDD name=%A

DEVHELP Major Code: 0X0006 Minor Code: 125 (0X007D)

Description (OS) DevHlp_RegisterBeep Pre-Invocation
Tracepoint Static tracepoint in DEVHELP.
Minor Code 125 (0X007D)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

PTDBeep entry point=%A

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 126 (0X007E)

Description (OS) DevHlp_Beep Pre-Invocation
Tracepoint Static tracepoint in DEVHELP.
Minor Code 126 (0X007E)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

Beep frequency=%W Beep duration=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 129 (0X0081)

Description (OS) DevHlp_ABIOSCall Post-Invocation
Tracepoint Static tracepoint in DEVHELP.
Minor Code 129 (0X0081)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

return code=%W
DEVHELP Major Code: 0X0006 Minor Code: 130 (0X0082)

Description: (OS) DevHlp_ABIOSCommonEnt Post-Invocation

Tracepoint: Static tracepoint in DEVHELP.

Minor Code: 130 (0X0082)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 131 (0X0083)

Description: (OS) DevHlp_ABIOSGetParms Post-Invocation

Tracepoint: Static tracepoint in DEVHELP.

Minor Code: 131 (0X0083)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 132 (0X0084)

Description: (OS) DevHlp_AddTraceEvent Post-Invocation

Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 132 (0X0084)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 133 (0X0085)

Description: (OS) DevHlp_AllocGDTSel Post-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 133 (0X0085)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 134 (0X0086)

Description: (OS) DevHlp_AllocPhys Post-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 134 (0X0086)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: Return Code=%W Physical Address=%D
<table>
<thead>
<tr>
<th>DEVHELP Major Code: 0X0006 Minor Code: 135 (0X0087)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>(OS) DevHlp_AllocReqPacket Post-Invocation</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td>135 (0X0087)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
<tr>
<td>return code=%W  request packet=%A</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEVHELP Major Code: 0X0006 Minor Code: 136 (0X0088)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>(OS) DevHlp_AttachDD Post-Invocation</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td>136 (0X0088)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
<tr>
<td>Return code=%W</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEVHELP Major Code: 0X0006 Minor Code: 137 (0X0089)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>(OS) DevHlp_Block Post-Invocation</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 138 (0X008A)

Description
(OS) DevHip_DeRegister Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
138 (0X008A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W  Number of monitors=%W

DEVHELP Major Code: 0X0006 Minor Code: 139 (0X008B)

Description
(OS) DevHip_DevDone Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
139 (0X008B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 140 (0X008C)
### DEVHELP Major Code: 0X0006 Minor Code: 140 (0X008C)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_EOI Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>140 (0X008C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%W</td>
</tr>
</tbody>
</table>

### DEVHELP Major Code: 0X0006 Minor Code: 141 (0X008D)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_FreeLIDEntry Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>141 (0X008D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%W</td>
</tr>
</tbody>
</table>

### DEVHELP Major Code: 0X0006 Minor Code: 142 (0X008E)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_FreePhys Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>142 (0X008E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 143 (0X008F)

Description
(OS) DevHlp_FreeReqPacket Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
143 (0X008F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 144 (0X0090)

Description
(OS) DevHlp_GetDeviceBlock Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
144 (0X0090)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return Code=%W Real Mode Pointer=%A Protect Mode Pointer=%A

DEVHELP Major Code: 0X0006 Minor Code: 145 (0X0091)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_GetDOSVar Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>145 (0X0091)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 146 (0X0092)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_GetLIDEntry Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>146 (0X0092)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%W LID=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 147 (0X0093)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_InternalError Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>147 (0X0093)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 148 (0X0094)

Description
(OS) DevHlp_Lock Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
148 (0X0094)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
return code=%W     lock handle=%D

DEVHELP Major Code: 0X0006 Minor Code: 149 (0X0095)

Description
(OS) DevHlp_LogEntry Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
149 (0X0095)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 150 (0X0096)

Description
(OS) DevHlp_MonCreate Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
150 (0x0096)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Return code=%%W Handle=%%W

--------------------------------------------

DEVHELP Major Code: 0x0006 Minor Code: 151 (0x0097)

**Description**
(OS) DevHlp_MonFlush Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
151 (0x0097)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Return code=%%W

--------------------------------------------

DEVHELP Major Code: 0x0006 Minor Code: 152 (0x0098)

**Description**
(OS) DevHlp_MonWrite Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
152 (0x0098)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 153 (0X0099)

Description
(OS) DevHlp_PhysToGDTSelector Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
153 (0X0099)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 154 (0X009A)

Description
(OS) DevHlp_PhysToUVirt Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
154 (0X009A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

return code=%W selector=%W offset=%W

DEVHELP Major Code: 0X0006 Minor Code: 155 (0X009B)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_PhysToVirt Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>155 (0X009B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 156 (0X009C)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_Profiling Kernel Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>156 (0X009C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 157 (0X009D)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_ProtToReal Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>157 (0X009D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 158 (0X009E)

**Description**
(OS) DevHip_PullParticular Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
158 (0X009E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 159 (0X009F)

**Description**
(OS) DevHip_PullReqPacket Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
159 (0X009F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

return code=%W   request packet=%A

DEVHELP Major Code: 0X0006 Minor Code: 160 (0X00A0)

**Description**
(OS) DevHlp_PushReqPacket Post-Invocation

**Tracepoint**: Static tracepoint in DEVHELP.

**Minor Code**: 160 (0X00A0)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**: No parameters traced.

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 161 (0X00A1)

**Description**: (OS) DevHlp_QueueFlush Post-Invocation

**Tracepoint**: Static tracepoint in DEVHELP.

**Minor Code**: 161 (0X00A1)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**: No parameters traced.

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 162 (0X00A2)

**Description**: (OS) DevHlp_QueueInit Post-Invocation

**Tracepoint**: Static tracepoint in DEVHELP.

**Minor Code**: 162 (0X00A2)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**: No parameters traced.
DEVHELP Major Code: 0X0006 Minor Code: 163 (0X00A3)

Description  
(OS) DevHip_QueueRead Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
163 (0X00A3)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
Return code=%W  Character read=%B

DEVHELP Major Code: 0X0006 Minor Code: 164 (0X00A4)

Description  
(OS) DevHip_QueueWrite Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
164 (0X00A4)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 165 (0X00A5)

Description  
(OS) DevHip_RealToProt Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.
DEVHELP Major Code: 0X0006 Minor Code: 166 (0X00A6)

Description: (OS) DevHlp_Register Post-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 166 (0X00A6)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 167 (0X00A7)

Description: (OS) DevHlp_RegStackUsage Post-Invocation
Tracepoint: Static tracepoint in DEVHELP.
Minor Code: 167 (0X00A7)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: Return code=%W
DEVHELP Major Code: 0X0006 Minor Code: 168 (0X00A8)

Description  
(OS) DevHip_ResetTimer Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
168 (0X00A8)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 170 (0X00AA)

Description  
(OS) DevHip_ROMCritSection Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
170 (0X00AA)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 171 (0X00AB)

Description  
(OS) DevHip_ProcRun Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
171 (0X00AB)
Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 172 (0X00AC)

Description
(OS) DevHlp_SchedClock Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
172 (0X00AC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 173 (0X00AD)

Description
(OS) DevHlp_SemClear Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
173 (0X00AD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 174 (0X00AE)
### (OS) DevHlp_SemHandle Post-Invocation

**Description**
(OS) DevHlp_SemHandle Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
174 (0X00AE)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Return code=%W     SemHandle=%D


### DEVHELP Major Code: 0X0006 Minor Code: 175 (0X00AF)

**Description**
(OS) DevHlp_SemRequest Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
175 (0X00AF)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Return code=%W


### DEVHELP Major Code: 0X0006 Minor Code: 176 (0X00B0)

**Description**
(OS) DevHlp_SendEvent Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
176 (0X00B0)

**Trace Groups**
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 177 (0X00B1)

Description
(OS) DevHlp_SetIRQ Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
177 (0X00B1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 178 (0X00B2)

Description
(OS) DevHlp_SetROMVector Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
178 (0X00B2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

DEVHELP Major Code: 0X0006 Minor Code: 179 (0X00B3)
### Description
(OS) DevHlp_SetTimer Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
179 (0X00B3)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 180 (0X00B4)

**Description**
(OS) DevHlp_SortReqPacket Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
180 (0X00B4)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 181 (0X00B5)

**Description**
(OS) DevHlp_TCYield Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
181 (0X00B5)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters
No parameters traced.

---------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 182 (0X00B6)

Description
(OS) DevHlp_TickCount Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
182 (0X00B6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

---------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 183 (0X00B7)

Description
(OS) DevHlp_Unlock Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
183 (0X00B7)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

---------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 184 (0X00B8)

Description
(OS) DevHip_UnPhysToVirt Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
184 (0X00B8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 185 (0X00B9)

**Description**
(OS) DevHip_UnSetIRQ Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
185 (0X00B9)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 186 (0X00BA)

**Description**
(OS) DevHip_VerifyAccess Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
186 (0X00BA)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 188 (0X00BC)

Description
(OS) DevHip_VirtToPhys Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
188 (0X00BC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Return code=%W  Physical Address=%D

DEVHELP Major Code: 0X0006 Minor Code: 189 (0X00BD)

Description
(OS) DevHip_VMAlloc Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
189 (0X00BD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 190 (0X00BE)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHip_VMFree Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>190 (0X00BE)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%D</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 191 (0X00BF)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHip_VMGlobalToProcess Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>191 (0X00BF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%D Linear address=%D</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 193 (0X00C1)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHip_VMLock Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>193 (0X00C1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
DEVHELP Major Code: 0X0006 Minor Code: 194 (0X00C2)

Description  
(OS) DevHlp_VMProcessToGlobal Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
194 (0X00C2)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
Return code=%D Linear address=%D

DEVHELP Major Code: 0X0006 Minor Code: 195 (0X00C3)

Description  
(OS) DevHlp_VMUnlock Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
195 (0X00C3)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 197 (0X00C5)
**Description**  
(OS) DevHlp_Yield Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
197 (0X00C5)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

Return code=%W

--------------------------------------------

**DEVHELP Major Code: 0X0006 Minor Code: 201 (0X00C9)**

**Description**  
(OS) DevHlp_FreeGDTSelector Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
201 (0X00C9)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

Return code=%W

--------------------------------------------

**DEVHELP Major Code: 0X0006 Minor Code: 202 (0X00CA)**

**Description**  
(OS) DevHlp_VirtToLin Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
202 (0X00CA)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.
DEVHELP Major Code: 0X0006 Minor Code: 203 (0X00CB)

Description
(OS) DevHip_LinToGDTSelector Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
203 (0X00CB)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 204 (0X00CC)

Description
(OS) DevHip_GetDescInfo Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
204 (0X00CC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Return code=%D
Attrib. byte=%B
Access byte=%B
Phys. address=%D
Descr. limit=%D

DEVHELP Major Code: 0X0006 Minor Code: 205 (0X00CD)
Description  (OS) DevHlp_LinToPageList Post-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  205 (0X00CD)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

Return code=%D

---------------------------------------------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 206 (0X00CE)

Description  (OS) DevHlp_PageListToLin Post-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  206 (0X00CE)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

Return code=%D Linear address=%D

---------------------------------------------------------------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 207 (0X00CF)

Description  (OS) DevHlp_PageListToGDTSelector Post-Invocation
Tracepoint  Static tracepoint in DEVHELP.
Minor Code  207 (0X00CF)
Trace Groups  No groups assigned.
DEVHELP Major Code: 0X0006 Minor Code: 209 (0X00D1)

Description
(OS) DevHlp_PhysToGDTSel Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
209 (0X00D1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 211 (0X00D3)

Description
(OS) DevHlp_AllocCtxHook Post-Invocation

Tracepoint
Static tracepoint in DEVHELP.

Minor Code
211 (0X00D3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
Return code=%W

DEVHELP Major Code: 0X0006 Minor Code: 212 (0X00D4)
Description (OS) DevHlp_ArmCtxHook Post-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 212 (0X00D4)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 213 (0X00D5)

Description (OS) DevHlp_FreeCtxHook Post-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 213 (0X00D5)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

Return code=%D

DEVHELP Major Code: 0X0006 Minor Code: 215 (0X00D7)

Description (OS) DevHlp_OpenEventSem Post-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 215 (0X00D7)

Trace Groups No groups assigned.
DEVHELP Major Code: 0X0006 Minor Code: 216 (0X00D8)

**Description**
(OS) DevHlp_CloseEventSem Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
216 (0X00D8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%D, Sem Handle=%D

DEVHELP Major Code: 0X0006 Minor Code: 217 (0X00D9)

**Description**
(OS) DevHlp_PostEventSem Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
217 (0X00D9)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%D, Sem Handle=%D

DEVHELP Major Code: 0X0006 Minor Code: 218 (0X00DA)
**DEVHELP Major Code: 0X0006 Minor Code: 218 (0X00DA)**

**Description**
(OS) DevHlp_ResetEventSem Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
218 (0X00DA)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
Return code=%%D, Sem Handle=%%D, Post Count=%%D

**DEVHELP Major Code: 0X0006 Minor Code: 221 (0X00DD)**

**Description**
(OS) DevHlp_ProcRun2 Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
221 (0X00DD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

**DEVHELP Major Code: 0X0006 Minor Code: 236 (0X00EC)**

**Description**
(OS) DevHlp_ModifyPriority Post-Invocation

**Tracepoint**
Static tracepoint in DEVHELP.

**Minor Code**
236 (0X00EC)

**Trace Groups**
No groups assigned.

**Trace Types**
DEVHELP Major Code: 0X0006 Minor Code: 237 (0X00ED)

Description  
(OS) DevHip_RegisterTmrDD Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
237 (0X00ED)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

pqwTmrRollover=%A pqwTmr=%A

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 238 (0X00EE)

Description  
(OS) DevHip_RegisterPerfCtrs Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
238 (0X00EE)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

Return code=%W

--------------------------------------------

DEVHELP Major Code: 0X0006 Minor Code: 252 (0X00FC)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_RegisterPDD Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>252 (0X00FC)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 253 (0X00FD)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_RegisterBeep Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>253 (0X00FD)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return code=%%W</td>
</tr>
</tbody>
</table>

DEVHELP Major Code: 0X0006 Minor Code: 254 (0X00FE)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DevHlp_Beep Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static tracepoint in DEVHELP.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>254 (0X00FE)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
Miscellaneous System Trace Events

The tracepoints for the Miscellaneous major code are identified in the following table. These tracepoints are listed separately from the other kernel tracepoints because they are static tracepoints. They are compiled with the code.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Disk device driver:

Trace events for DISK02 Major Code: 0X0007, sorted by minor code.
Trace events for DISK02 Major Code: 0X0007, sorted by tracepoint.

Trace Events for DISK02 Major Code: 0X0007, Sorted by Minor Code

00001 (0X0001)  (OS) Disk Device Driver Read Pre-Invocation
00002 (0X0002)  (OS) Disk Device Driver Write Pre-Invocation
00003 (0X0003)  (OS) Disk Device Driver WriteVerify Pre-Invocation
00006 (0X0006)  (OS) Disk Device Driver SCB Transfer Pre-Invocation
32769 (0X8001)  (OS) Disk Device Driver Read Post-Invocation
32770 (0X8002)  (OS) Disk Device Driver Write Post-Invocation
32771 (0X8003)  (OS) Disk Device Driver WriteVerify Post-Invocation
32774 (0X8006)  (OS) Disk Device Driver SCB Transfer Post-Invocation

Trace Events for DISK02 Major Code: 0X0007, Sorted by Tracepoint

(OS) Disk Device Driver Read Post-Invocation 32769 (0X8001)
(OS) Disk Device Driver Read Pre-Invocation 00001 (0X0001)
(OS) Disk Device Driver SCB Transfer Post-Invocation 32774 (0X8006)
(OS) Disk Device Driver SCB Transfer Pre-Invocation 00006 (0X0006)
(OS) Disk Device Driver Write Post-Invocation 32770 (0X8002)
(OS) Disk Device Driver Write Pre-Invocation 00002 (0X0002)
(OS) Disk Device Driver WriteVerify Post-Invocation 32771 (0X8003)
(OS) Disk Device Driver WriteVerify Pre-Invocation 00003 (0X0003)

DISK02 Major Code: 0X0007 Minor Code: 1 (0X0001)
**Description** (OS) Disk Device Driver Read Pre-Invocation

**Tracepoint** Static tracepoint in DISK02.

**Minor Code** 1 (0X0001)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Drive ID=%W Relative Block Addr=%Q
Number of Sectors=%W Buffer Addr=%D

--------------------------------------------

**DISK02 Major Code: 0X0007 Minor Code: 2 (0X0002)**

**Description** (OS) Disk Device Driver Write Pre-Invocation

**Tracepoint** Static tracepoint in DISK02.

**Minor Code** 2 (0X0002)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Drive ID=%W Relative Block Addr=%Q
Number of Sectors=%W Buffer Addr=%D

--------------------------------------------

**DISK02 Major Code: 0X0007 Minor Code: 3 (0X0003)**

**Description** (OS) Disk Device Driver WriteVerify Pre-Invocation

**Tracepoint** Static tracepoint in DISK02.

**Minor Code**
Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters

- Drive ID=%W Relative Block Addr=%Q
- Number of Sectors=%W Buffer Addr=%D

DISK02 Major Code: 0X0007 Minor Code: 6 (0X0006)

Description  
(OS) Disk Device Driver SCB Transfer Pre-Invocation

Tracepoint  
Static tracepoint in DISK02.

Minor Code  
6 (0X0006)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters

- Drive ID=%W Relative Block Addr=%Q
- Number of Sectors=%W
- Req Offset=%W Req Selector=%W Req List Entry Offset=%D
- Scatter/Gather Count=%W Command=%W

DISK02 Major Code: 0X0007 Minor Code: 32769 (0X8001)

Description  
(OS) Disk Device Driver Read Post-Invocation

Tracepoint  
Static tracepoint in DISK02.

Minor Code  
32769 (0X8001)

Trace Groups  
No groups assigned.

Trace Types
DISK02 Major Code: 0X0007 Minor Code: 32770 (0X8002)

Description  (OS) Disk Device Driver Write Post-Invocation
Tracepoint  Static tracepoint in DISK02.
Minor Code  32770 (0X8002)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

-----

DISK02 Major Code: 0X0007 Minor Code: 32771 (0X8003)

Description  (OS) Disk Device Driver WriteVerify Post-Invocation
Tracepoint  Static tracepoint in DISK02.
Minor Code  32771 (0X8003)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

-----

DISK02 Major Code: 0X0007 Minor Code: 32774 (0X8006)
Description: (OS) Disk Device Driver SCB Transfer Post-Invocation

Tracepoint: Static tracepoint in DISK02.

Minor Code: 32774 (0X8006)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

Drive ID=%W Completion Status=%W

DASD manager device driver:

Trace events for OS2DASD Major Code: 0X0007, sorted by minor code.
Trace events for OS2DASD Major Code: 0X0007, sorted by tracepoint.

--------------------------------------------

Trace Events for OS2DASD Major Code: 0X0007, Sorted by Minor Code

00008 (0X0008) (OS) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation
00009 (0X0009) (OS) DASD Manager IOCTL Pre-Invocation
00010 (0X000A) (OS) DASD Manager Strategy-2 Request List Header Pre-Invocation
00011 (0X000B) (OS) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation
00012 (0X000C) (OS) DASD Manager IORB Pre-Invocation
00136 (0X0088) (OS) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation
00137 (0X0089) (OS) DASD Manager IOCTL Post-Invocation
00138 (0X008A) (OS) DASD Manager Strategy-2 Request List Header Post-Invocation
00139 (0X008B) (OS) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation
00140 (0X008C) (OS) DASD Manager IORB Post-Invocation

--------------------------------------------

Trace Events for OS2DASD Major Code: 0X0007, Sorted by Tracepoint

(0S) DASD Manager IOCTL Post-Invocation 00137 (0X0089)
(0S) DASD Manager IOCTL Pre-Invocation 00009 (0X0009)
(0S) DASD Manager IORB Post-Invocation 00140 (0X008C)
(0S) DASD Manager IORB Pre-Invocation 00012 (0X000C)
(0S) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation 00136 (0X0088)
(0S) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation 00008 (0X0008)
(0S) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation 00139 (0X008B)
(0S) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation 00011 (0X000B)
(0S) DASD Manager Strategy-2 Request List Header Post-Invocation 00138 (0X008A)
(0S) DASD Manager Strategy-2 Request List Header Pre-Invocation 00010 (0X000A)
OS2DASD Major Code: 0X0007 Minor Code: 8 (0X0008)

**Description**
(OS) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation

**Tracepoint**
Static tracepoint in OS2DASD.

**Minor Code**
8 (0X0008)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
\[ pRequest=%A \quad Unit=%B \quad %I1 \quad Drive=%S \quad Cmd=%B \quad %I1 \quad %S \quad %I3 \quad Prty=%B \quad %I1 \\
\quad cSGList=%W \quad RBA=%F \quad Sectors=%F \quad %I4 \]

OS2DASD Major Code: 0X0007 Minor Code: 9 (0X0009)

**Description**
(OS) DASD Manager IOCTL Pre-Invocation

**Tracepoint**
Static tracepoint in OS2DASD.

**Minor Code**
9 (0X0009)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
\[ pRequest=%A \quad Unit=%B \quad %I1 \quad Drive=%S \quad Cat/Func=%B \quad %B \quad %S \quad %I3 \quad Prty=%B \quad %I1 \\
\quad cSGList=%W \quad CH=%W \quad %W \quad Sectors=%F \quad %I4 \]

OS2DASD Major Code: 0X0007 Minor Code: 10 (0X000A)

**Description**
(OS) DASD Manager Strategy-2 Request List Header Pre-Invocation
### Tracepoint

Static tracepoint in OS2DASD.

### Minor Code

10 (0X000A)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

\[ pRLH=%A \text{ Count}=\%W \text{ Unit}=\%B \%\text{i1} \text{ Drive}=\%S \text{ Ctrl}=\%W \]

--------------------------------------------

OS2DASD Major Code: 0X0007 Minor Code: 11 (0X000B)

### Description

(OS) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation

### Tracepoint

Static tracepoint in OS2DASD.

### Minor Code

11 (0X000B)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

\[ p\text{Request}=\%A \text{ Unit}=\%B \%\text{i1} \text{ Drive}=\%S \text{ Cmd}=\%B \%\text{i1} \%\text{i1} \text{ Ctrl}=\%W \text{ Prty}=\%B \%\text{i1} \]
\[ c\text{SGList}=\%W \text{ RBA}=\%F \text{ Sectors}=\%F \text{ pRLH}=\%W \%\text{i2} \]

--------------------------------------------

OS2DASD Major Code: 0X0007 Minor Code: 12 (0X000C)

### Description

(OS) DASD Manager IORB Pre-Invocation

### Tracepoint

Static tracepoint in OS2DASD.

### Minor Code

12 (0X000C)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.
Traced Parameters

pRequest=%A Unit=%B %I1 %I2 Cmd=%B %S %I1 Ctrl=%W %I2

cSGList=%W RBA=%F Sectors=%F %I4

--------------------------------------------

OS2DASD Major Code: 0X0007 Minor Code: 136 (0X0088)

Description
(OS) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation

Tracepoint
Static tracepoint in OS2DASD.

Minor Code
136 (0X0088)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

pRequest=%A Status=%B %I1 ErrorCode=%B %I1 SectorsDone=%F

--------------------------------------------

OS2DASD Major Code: 0X0007 Minor Code: 137 (0X0089)

Description
(OS) DASD Manager IOCTL Post-Invocation

Tracepoint
Static tracepoint in OS2DASD.

Minor Code
137 (0X0089)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

pRequest=%A Status=%B %I1 ErrorCode=%B %I5

--------------------------------------------

OS2DASD Major Code: 0X0007 Minor Code: 138 (0X008A)
OS2DASD Major Code: 0X0007 Minor Code: 138 (0X008A)

Description
(OS) DASD Manager Strategy-2 Request List Header Post-Invocation

Tracepoint
Static tracepoint in OS2DASD.

Minor Code
138 (0X008A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
pRLH=%A DoneCount=%W Status=%B %I1

OS2DASD Major Code: 0X0007 Minor Code: 139 (0X008B)

Description
(OS) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation

Tracepoint
Static tracepoint in OS2DASD.

Minor Code
139 (0X008B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
pRequest=%A Status=%B %I1 ErrorCode=%B %I1 SectorsDone=%F

OS2DASD Major Code: 0X0007 Minor Code: 140 (0X008C)

Description
(OS) DASD Manager IORB Post-Invocation

Tracepoint
Static tracepoint in OS2DASD.

Minor Code
140 (0X008C)

Trace Groups
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
pRequest=%A Status=%W ErrorCode=%W SectorsDone=%F

--------------------------------------------

Resource Manager Device Driver Trace Events

The tracepoints for the Miscellaneous major code are identified in the following table. These tracepoints are listed separately from the other kernel tracepoints because they are static tracepoints. They are compiled with the code.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Resource manager device driver:

Trace events for RESOURCE major code: 0X0008, sorted by minor code.
Trace events for RESOURCE major code: 0X0008 ,sorted by tracepoint.

--------------------------------------------

Trace Events for RESOURCE Major Code: 0X0008, Sorted by Minor Code

00001 (0X0001) (OS) Resource Manager Allocate
00002 (0X0002) (OS) Resource Manager Allocate
00003 (0X0003) (OS) Resource Manager Allocate
00017 (0X0011) (OS) Resource Manager Deallocate
00018 (0X0012) (OS) Resource Manager Deallocate
00019 (0X0013) (OS) Resource Manager Deallocate

--------------------------------------------

Trace Events for RESOURCE Major Code: 0X0008, Sorted by Tracepoint

(OS) Resource Manager Deallocate 00017 (0X0011)
(OS) Resource Manager Deallocate 00018 (0X0012)
(OS) Resource Manager Deallocate 00019 (0X0013)
(OS) Resource Manager Allocate 00001 (0X0001)
(OS) Resource Manager Allocate 00002 (0X0002)
(OS) Resource Manager Allocate 00003 (0X0003)

--------------------------------------------

RESOURCE Major Code: 0X0008 Minor Code: 1 (0X0001)
**Description**
(OS) Resource Manager Allocate

**Tracepoint**
Static trace point in RESOURCE.

**Minor Code**
1 (0X0001)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Driver=%s
%s=%w %w Flags=%s Rc=%w

--------------------------------------------

**RESOURCE Major Code: 0X0008 Minor Code: 2 (0X0002)**

**Description**
(OS) Resource Manager Allocate

**Tracepoint**
Static trace point in RESOURCE.

**Minor Code**
2 (0X0002)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

Driver=%s
%s=%w Flags=%s Rc=%w

--------------------------------------------

**RESOURCE Major Code: 0X0008 Minor Code: 3 (0X0003)**

**Description**
(OS) Resource Manager Allocate

**Tracepoint**
Static trace point in RESOURCE.

**Minor Code**
RESOURCE Major Code: 0X0008 Minor Code: 17 (0X0011)

Description: (OS) Resource Manager Deallocate
Tracepoint: Static trace point in RESOURCE.
Minor Code: 17 (0X0011)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
    Driver=%s
    %s=%d%f Flags=%s Rc=%w

-------------------------------

RESOURCE Major Code: 0X0008 Minor Code: 18 (0X0012)

Description: (OS) Resource Manager Deallocate
Tracepoint: Static trace point in RESOURCE.
Minor Code: 18 (0X0012)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
    Driver=%s
    %s=%d%f Flags=%s Rc=%w

-------------------------------
Driver=%s
%s=%w Flags=%s Rc=%w

--------------------------------------------

RESOURCE Major Code: 0X0008 Minor Code: 19 (0X0013)

Description
(OS) Resource Manager Deallocate

Tracepoint
Static trace point in RESOURCE.

Minor Code
19 (0X0013)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

Driver=%s
%s=%f%f Flags=%s Rc=%w

--------------------------------------------

DOSCALL1.DLL Trace Events

The tracepoints for the DOSCALL1.DLL kernel services major code are identified in the following tables. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for DOSCALL1 Major Code: 0X0010, sorted by minor code.
Trace events for DOSCALL1 Major Code: 0X0010 ,sorted by tracepoint.
Kernel API Tracepoints Indirected Via DOSCALL1,
QUECALLS API Tracepoints Indirected Via DOSCALL1.

--------------------------------------------

Trace Events for DOSCALL1 Major Code: 0X0010, Sorted by Minor Code

00008 (0X0008) (OS) DosFSRamSemClear Pre-Invocation
00009 (0X0009) (OS) DosFSRamSemRequest Pre-Invocation
00014 (0X000E) (OS) DosReadAsync Pre-Invocation
00015 (0X000F) (OS) DosScanEnv Pre-Invocation
Trace Events for DOSCALL1 Major Code: 0X0010, Sorted by Tracepoint

DOS32EXCEPTIONCALLBACK 00362 (0X016A)
DOS32R3EXCEPTIONDISPATCHER 00361 (0X0169)
DOS32RAISEEXCEPTION 00035 (0X0023)
DOS32SETEXCEPTIONHANDLER 00034 (0X0022)
Dos32SubAlloc Post-Invocation 32792 (0X8018)
Dos32SubAlloc Pre-Invocation 00024 (0X0018)
Dos32SubFree Post-Invocation 32794 (0X801A)
Dos32SubFree Pre-Invocation 00026 (0X001A)
Dos32SubSet Post-Invocation 32796 (0X801C)
Dos32SubSet Pre-Invocation 00028 (0X001C)
DOS32UNSETEXCEPTIONHANDLER 00036 (0X0024)
DOS32UNWINDEXCEPTION 00037 (0X0025)
DOSFSRAMSEMCLEAR 00008 (0X0008)
DOSFSRAMSEMREQUEST 00009 (0X0009)
DOSREADASYNC 00014 (0X000E)
DOSSCANENV 00015 (0X000F)
DOSEXCHGEFFECT 00016 (0X0010)
DOSSEMCLEAR 00017 (0X0011)
DOSSEMREQUEST 00018 (0X0012)
DOSSEMSET 00019 (0X0013)
DOSSEMWAIT 00020 (0X0014)
DOSSETCP 00021 (0X0015)
DOSSUBSET 00022 (0X0016)
DOSWRITEASYNC 00023 (0X001B)
POSTGETMSG 33025 (0X8101)
POSTINSMSG 33026 (0X8102)
POSTPUTMSG 33027 (0X8103)
PREGETMSG 00257 (0X0101)
PREINMSG 00258 (0X0102)
PREPUTMSG 00259 (0X0103)
postDOS32SETEXCEPTIONHANDLER 32802 (0X8022)
postDOS32UNSETEXCEPTIONHANDLER 32804 (0X8024)
poldtDOSERRCLASS 32800 (0X8020)
poldtDOSFSRAMSEMCLEAR 32776 (0X8008)
poldtDOSFSRAMSEMREQUEST 32777 (0X8009)
poldtDOSQAPPTYPE 32901 (0X8021)
poldtDOSREADASYNC 32782 (0X800E)
poldtDOSSCANENV 32783 (0X800F)
poldtDOSEXCHGEFFECT 32784 (0X8010)
poldtDOSSEMCLEAR 32785 (0X8011)
poldtDOSSEMREQUEST 32786 (0X8012)
poldtDOSSEMSET 32787 (0X8013)
poldtDOSSEMWAIT 32788 (0X8014)
poldtDOSSUBSET 32789 (0X8015)
poldtDOSSUBPROC 32790 (0X8016)
poldtDOSSUBALLOC 32791 (0X8017)
poldtDOSSUBFREE 32793 (0X8019)
poldtDOSSUBSET 32795 (0X801B)
poldtDOSWRITEASYNC 32798 (0X801E)
predeDOSERRCLASS 00032 (0X0020)
predeDOSQAPPTYPE 00033 (0X0021)
predeDOSSUBALLOC 00023 (0X0017)
predeDOSSUBFREE 00025 (0X0019)
xceptExecuteUserExceptionHandler 00363 (0X016B)
UniThunk 16:32 Return33029 (0X8105)
UniThunk 32:16 Return33028 (0X8104)

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 8 (0X0008)

Description
(OS) DosFSRamSemClear Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOSFSRAMSEMCLEAR

Minor Code
8 (0X0008)

Trace Groups
SEM
Trace Types
PRE

Traced Parameters
Sem Ptr = %P %A

Note: Parameters are trace with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

---------------------------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 9 (0X0009)

Description
(OS) DosFSRamSemRequest Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOSFSRAMSEMREQUEST

Minor Code
9 (0X0009)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Timeout = %P %F Sem Ptr = %P %A

Note: Parameters are trace with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

---------------------------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 14 (0X000E)

Description
(OS) DosReadAsync Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOSREADASYNC

Minor Code
14 (0X000E)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
Sem at = %A Handle = %W
DOSCALL1 Major Code: 0X0010 Minor Code: 15 (0X000F)

Description  
(OS) DosScanEnv Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: DOSCALL1.DOSSCANENV

Minor Code  
15 (0X000F)

Trace Groups  
LNK

Trace Types  
PRE

Traced Parameters  
Current Env = %S  
Env Pointer = %A

DOSCALL1 Major Code: 0X0010 Minor Code: 16 (0X0010)

Description  
(OS) DosSearchPath Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEARCHPATH

Minor Code  
16 (0X0010)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
Control = %W  
Path = %S  
File = %S

DOSCALL1 Major Code: 0X0010 Minor Code: 17 (0X0011)
**Description**  (OS) DosSemClear Pre-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMCLEAR

**Minor Code**  17 (0X0011)

**Trace Groups**  SEM

**Trace Types**  PRE

**Traced Parameters**  
Sem Handle = %D

--------------------------------------------

**Description**  (OS) DosSemRequest Pre-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMREQUEST

**Minor Code**  18 (0X0012)

**Trace Groups**  SEM

**Trace Types**  PRE

**Traced Parameters**  
Sem Handle = %D

--------------------------------------------

**Description**  (OS) DosSemSet Pre-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMSET

**Minor Code**  19 (0X0013)

**Trace Groups**  SEM

**Trace Types**  PRE
Traced Parameters
Sem Handle = %D

DOSCALL1 Major Code: 0X0010 Minor Code: 20 (0X0014)

Description
(OS) DosSemWait Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMWAIT

Minor Code
20 (0X0014)

Trace Groups
SEM

Trace Types
PRE

Traced Parameters
Sem Handle = %D

DOSCALL1 Major Code: 0X0010 Minor Code: 21 (0X0015)

Description
(OS) DosSetCp Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOSSETCP

Minor Code
21 (0X0015)

Trace Groups
NLS

Trace Types
PRE

Traced Parameters
Code Page Id= %W

DOSCALL1 Major Code: 0X0010 Minor Code: 22 (0X0016)
**Description**  
(OS) DosSetProcCp Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: DOSCALL1.DOSSETPROCCP

**Minor Code**  
22 (0X0016)

**Trace Groups**  
NLS

**Trace Types**  
PRE

**Traced Parameters**  
Code Page Id= %W

--------------------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 23 (0X0017)**

**Description**  
(OS) DosSubAlloc Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: DOSCALL1.preDOSSUBALLOC

**Minor Code**  
23 (0X0017)

**Trace Groups**  
MSP

**Trace Types**  
PRE

**Traced Parameters**  
Selector = %W Size = %W

--------------------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 24 (0X0018)**

**Description**  
(OS) Dos32SubAlloc Pre-Invocation

**Tracepoint**  
Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

**Minor Code**  
24 (0X0018)

**Trace Groups**  
MSP

**Trace Types**  
PRE
Traced Parameters

pHdr = %F Size = %D

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 25 (0X0019)

Description
(OS) DosSubFree Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.preDOSSUBFREE

Minor Code
25 (0X0019)

Trace Groups
MSP

Trace Types
PRE

Traced Parameters

Selector = %W Offset = %W
Size = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 26 (0X001A)

Description
(OS) Dos32SubFree Pre-Invocation

Tracepoint
Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

Minor Code
26 (0X001A)

Trace Groups
MSP

Trace Types
PRE

Traced Parameters

pHdr = %F Offset = %F
Size = %D

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 27 (0X001B)
Description: (OS) DosSubSet Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: DOSCALL1.DOSSUBSET

Minor Code: 27 (0X001B)

Trace Groups: MSP

Trace Types: PRE

Traced Parameters:

Selector = %W Flags = %W
Size = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 28 (0X001C)

Description: (OS) Dos32SubSet Pre-Invocation

Tracepoint: Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

Minor Code: 28 (0X001C)

Trace Groups: MSP

Trace Types: PRE

Traced Parameters:

pHdr = %F Flags = %D
Size = %D

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 30 (0X001E)

Description: (OS) DosWriteAsync Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: DOSCALL1.DOSWRITEASYNC

Minor Code:
30 (0X001E)

**Trace Groups**
- FS

**Trace Types**
- PRE

**Traced Parameters**
- Sem at = %A Handle = %W

------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 32 (0X0020)**

**Description**
(OS) DosErrClass Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.preDOSERRCLASS

**Minor Code**
32 (0X0020)

**Trace Groups**
- FS

**Trace Types**
- PRE

**Traced Parameters**
- Error Code = %W

------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 33 (0X0021)**

**Description**
(OS) DosQAppType Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.preDOSQAPPTYPE

**Minor Code**
33 (0X0021)

**Trace Groups**
- LDR

**Trace Types**
- PRE

**Traced Parameters**
- File = %S
DOSCALL1 Major Code: 0X0010 Minor Code: 34 (0X0022)

**Description**
(OS) Dos32SetExceptionHandler Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.DOS32SETEXCEPTIONHANDLER

**Minor Code**
34 (0X0022)

**Trace Groups**
EXMG

**Trace Types**
PRE

**Traced Parameters**

ExcHandlerStructAddress = %D

DOSCALL1 Major Code: 0X0010 Minor Code: 35 (0X0023)

**Description**
(OS) Dos32RaiseException Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.DOS32RAISEEXCEPTION

**Minor Code**
35 (0X0023)

**Trace Groups**
EXMG

**Trace Types**
PRE

**Traced Parameters**

Exception Report Record:
Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F
Exception Address=%F Parameter Count=%F
P1=%F P2=%F P3=%F P4=%F

**Note:** Parameters are traced only with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

DOSCALL1 Major Code: 0X0010 Minor Code: 36 (0X0024)
**Description** (OS) Dos32UnsetExceptionHandler Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: DOSCALL1.DOS32UNSETEXCEPTIONHANDLER

**Minor Code** 36 (0X0024)

**Trace Groups** EXMG

**Trace Types** PRE

**Traced Parameters**

ExcHandlerStructAddress = %D

-------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 37 (0X0025)**

**Description** (OS) Dos32UnwindException Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: DOSCALL1.DOS32UNWINDEXCEPTION

**Minor Code** 37 (0X0025)

**Trace Groups** EXMG

**Trace Types** PRE

**Traced Parameters**

ExcHandlerStructAddress = %D

-------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 257 (0X0101)**

**Description** (OS) DosGetMessage Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: DOSCALL1.PREGETMSG

**Minor Code** 257 (0X0101)

**Trace Groups** MSG
Trace Types
PRE

Traced Parameters

MsgNumber = %W
Filename = %S

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 258 (0X0102)

Description
(OS) DosInsMessage Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.PREINSMSG

Minor Code
258 (0X0102)

Trace Groups
MSG

Trace Types
PRE

Traced Parameters

IV Count = %W
Length of Input Message %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 259 (0X0103)

Description
(OS) DosPutMessage Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.PREPUTMSG

Minor Code
259 (0X0103)

Trace Groups
MSG

Trace Types
PRE

Traced Parameters

File Handle = %W
Message Length = %W
DOSCALL1 Major Code: 0X0010 Minor Code: 32776 (0X8008)

**Description**
(OS) DosFSRamSemClear Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSFSRAMSEMCLEAR

**Minor Code**
32776 (0X8008)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32777 (0X8009)

**Description**
(OS) DosFSRamSemRequest Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSFSRAMSEMREQUEST

**Minor Code**
32777 (0X8009)

**Trace Groups**
SEM

**Trace Types**
POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32782 (0X800E)
**Description**
(OS) DosReadAsync Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSREADASYNC

**Minor Code**
32782 (0X800E)

**Trace Groups**
FS

**Trace Types**
POST

**Traced Parameters**

Bytes Read = %W Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32783 (0X800F)

**Description**
(OS) DosScanEnv Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSCANENV

**Minor Code**
32783 (0X800F)

**Trace Groups**
LNK

**Trace Types**
POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32784 (0X8010)

**Description**
(OS) DosSearchPath Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEARCHPATH

**Minor Code**
32784 (0X8010)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32785 (0X8011)

**Description**

(OS) DosSemClear Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMCLEAR

**Minor Code**

32785 (0X8011)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32786 (0X8012)

**Description**

(OS) DosSemRequest Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMREQUEST

**Minor Code**

32786 (0X8012)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**
DOSCALL1 Major Code: 0X0010 Minor Code: 32787 (0X8013)

Description: (OS) DosSemSet Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMSET
Minor Code: 32787 (0X8013)
Trace Groups: SEM
Trace Types: POST
Traced Parameters:

Return Code = %W

DOSCALL1 Major Code: 0X0010 Minor Code: 32788 (0X8014)

Description: (OS) DosSemWait Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMSET
Minor Code: 32788 (0X8014)
Trace Groups: SEM
Trace Types: POST
Traced Parameters:

Return Code = %W

DOSCALL1 Major Code: 0X0010 Minor Code: 32789
(Ox8015)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSetCp Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSETCP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32789 (Ox8015)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>NLS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code= %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32790 (Ox8016)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSetProcCp Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSETPROCCP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32790 (Ox8016)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>NLS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code= %W</td>
</tr>
</tbody>
</table>

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32791 (Ox8017)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSubAlloc Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td></td>
</tr>
</tbody>
</table>
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBALLOC

**Minor Code**
32791 (0X8017)

**Trace Groups**
MSP

**Trace Types**
POST

**Traced Parameters**
Offset = %W Return Code = %W

DOSCALL1 Major Code: 0X0010 Minor Code: 32792 (0X8018)

**Description**
(OS) Dos32SubAlloc Post-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

**Minor Code**
32792 (0X8018)

**Trace Groups**
MSP

**Trace Types**
POST

**Traced Parameters**
Offset = %F Return Code = %D

DOSCALL1 Major Code: 0X0010 Minor Code: 32793 (0X8019)

**Description**
(OS) DosSubFree Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBFREE

**Minor Code**
32793 (0X8019)

**Trace Groups**
MSP

**Trace Types**
POST
Traced Parameters

Return Code = %W

---------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32794 (0X801A)

Description
(OS) Dos32SubFree Post-Invocation

Tracepoint
Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

Minor Code
32794 (0X801A)

Trace Groups
MSP

Trace Types
POST

Traced Parameters

Return Code = %D

---------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32795 (0X801B)

Description
(OS) DosSubSet Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBSET

Minor Code
32795 (0X801B)

Trace Groups
MSP

Trace Types
POST

Traced Parameters

Return Code = %W

---------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32796
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) Dos32SubSet Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32796 (0x801C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MSP</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %D</td>
</tr>
</tbody>
</table>

DOSCALL1 Major Code: 0X0010 Minor Code: 32798 (0x801E)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosWriteAsync Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: DOSCALL1.postDOSWRITEASYNC</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32798 (0x801E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Bytes Written = %W Return Code = %W</td>
</tr>
</tbody>
</table>

DOSCALL1 Major Code: 0X0010 Minor Code: 32800 (0x8020)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosErrClass Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td></td>
</tr>
</tbody>
</table>
Public symbol defined dynamic tracepoint: DOSCALL1.postDOSERRCLASS

**Minor Code**

32800 (0X8020)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Class = %W Action = %W
Locus = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32801 (0X8021)

**Description**

(OS) DosQAppType Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSQAPPTYPE

**Minor Code**

32801 (0X8021)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Type = %W Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32802 (0X8022)

**Description**

(OS) Dos32SetExceptionHandler Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOS32SETEXCEPTIONHANDLER

**Minor Code**

32802 (0X8022)

**Trace Groups**

EXMG
Trace Types
POST

Traced Parameters
No parameters traced.

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 32804 (0X8024)

Description
(OS) Dos32UnsetExceptionHandler Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.postDOS32UNSETEXCEPTIONHANDLER

Minor Code
32804 (0X8024)

Trace Groups
EXMG

Trace Types
POST

Traced Parameters
Return Code = %D

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 33025 (0X8101)

Description
(OS) DosGetMessage Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.POSTGETMSG

Minor Code
33025 (0X8101)

Trace Groups
MSG

Trace Types
POST

Traced Parameters
Return Code = %W
Message Length = %W

---------------------------------------------
DOSCALL1 Major Code: 0X0010 Minor Code: 33026 (0X8102)

Description
(OS) DosInsMessage Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.POSTINSMSG

Minor Code
33026 (0X8102)

Trace Groups
MSG

Trace Types
POST

Traced Parameters
Return Code = %W
Message Length = %W

DOSCALL1 Major Code: 0X0010 Minor Code: 33027 (0X8103)

Description
(OS) DosPutMessage Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.POSTPUTMSG

Minor Code
33027 (0X8103)

Trace Groups
MSG

Trace Types
POST

Traced Parameters
Return Code = %W

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 361 (0X0169)
Description
(OS) Dos32R3ExceptionDispatcher Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOS32R3EXCEPTIONDISPATCHER

Minor Code
361 (0X0169)

Trace Groups
EXMG

Trace Types
PRE, API

Traced Parameters

Return address=%P %F Trap Number=%F pRepRec=%F pContext=%F

Exception Report Record:
Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F
Exception Address=%F Parameter Count=%F
P1=%F P2=%F P3=%F P4=%F

Context Record:
ContextFlags=%P %F
FPU_CNTRL=%W %I2 FPU_STATUS=%W %I2 FPU_TAG=%W %I2
FPU_IP=%F FPU_CS=%W FPU_OPCODE=%W FPU_DATAOFFS=%F %I4

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:
R0=%F %F %W R1=%F %F %W
R2=%F %F %W R3=%F %F %W
R4=%F %F %W R5=%F %F %W
R6=%F %F %W R7=%F %F %W
GS=%W %I2 FS=%W %I2 ES=%W %I2 DS=%W %I2
EDI=%F ESI=%F EAX=%F EBX=%F ECX=%F EDX=%F
EBP=%F EIP=%F CS=%W %I2 EFLAGS=%F ESP=%F SS=%W %I2

Note:
This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 362 (0X016A)

Description
(OS) Dos32ExceptionCallback Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.DOS32EXCEPTIONCALLBACK
Minor Code: 362 (0X016A)

Trace Groups: EXMG

Trace Types: PRE, API

Traced Parameters:

Return address=%P %F Trap Number=%F pRepRec=%F pContext=%F Disposition=%F

Exception Report Record:

Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F

Exception Address=%F Parameter Count=%F

P1=%F P2=%F P3=%F P4=%F

Context Record:

ContextFlags=%P %F

FPU_CNTRL=W %W FPU_STATUS=W %W FPU_TAG=W %W

FPU_IP=W FPU_CS=W FPU_OPCODE=W FPU_DATAOFFS=W

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:

R0=R %F %W R1=R %F %W

R2=R %F %W R3=R %F %W

R4=R %F %W R5=R %F %W

R6=R %F %W R7=R %F %W

GS=W %W FS=W %W ES=W %W DS=W %W

EDI=E %F ESI=E %F EAX=E %F EBX=E %F ECX=E %F EDX=E %F

EBP=E %F EIP=E %F CS=W %W EFLAGS=E %F ESP=E %F SS=W %W

Note:

This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

-------------------------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 363 (0X016B)

Description: (OS) xcptExecuteUserExceptionHandler Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: DOSCALL1.xcptExecuteUserExceptionHandler

Minor Code: 363 (0X016B)

Trace Groups: EXMG
**Trace Types**

PRE, INT

**Traced Parameters**

Return address=%P %F pRepRec=%F pRegRec=%F pContext=%F pDisContext=%F pNestCatcher=%F

Exception Report Record:

Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F

Exception Address=%F Parameter Count=%F

P1=%F P2=%F P3=%F P4=%F

Registration Record:

pNextRegRec=%P %F pHandler=%F

Context Record:

ContextFlags=%P %F

FPU_CNTRL=%W %I2 FPU_STATUS=%W %I2 FPU_TAG=%W %I2

FPU_IP=%F FPU_CS=%W FPU_OPCODE=%W FPU_DATAOFFS=%F %I4

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:

R0=%F %F %W R1=%F %F %W

R2=%F %F %W R3=%F %F %W

R4=%F %F %W R5=%F %F %W

R6=%F %F %W R7=%F %F %W

GS=%W %I2 FS=%W %I2 ES=%W %I2 DS=%W %I2

EDI=%F ESI=%F EAX=%F EBX=%F ECX=%F EDX=%F

EBP=%F EIP=%F CS=%W %I2 EFLAGS=%F ESP=%F SS=%W %I2

**Note:**

This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

--------------------------------------------

**DOSCALL1 Major Code: 0X0010 Minor Code: 33028 (0X8104)**

**Description**

(OS) UniThunk 32:16 Return

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.UT16_RETURN

**Minor Code**

33028 (0X8104)

**Trace Groups**

UT
Trace Types
POST, INT

Traced Parameters
Return address = %P %A

Note:
This tracepoint is available with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10 or later.

--------------------------------------------

DOSCALL1 Major Code: 0X0010 Minor Code: 33029 (0X8105)

Description
(OS) UniThunk 16:32 Return

Tracepoint
Public symbol defined dynamic tracepoint: DOSCALL1.UT32_RETURN

Minor Code
33029 (0X8105)

Trace Groups
UT

Trace Types
POST, INT

Traced Parameters
Return address = %P %F

Note:
This tracepoint is available with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10 or later.

--------------------------------------------

Kernel API Tracepoints Indirected Via DOSCALL1

The following table lists pre-invocation tracepoints for Kernel APIs that are indirected via DOSCALL1. These should be trace in conjunction with their corresponding Kernel pre-invocation tracepoint.

<table>
<thead>
<tr>
<th>DOSCALL1 API</th>
<th>Minor code</th>
<th>Group</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS32WAITCHILD</td>
<td>262</td>
<td>TK</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32BEEP</td>
<td>263</td>
<td>IO</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32PHYSICALDISK</td>
<td>264</td>
<td>FS</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32SETCP</td>
<td>265</td>
<td>TK</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32SETPROCESSCP</td>
<td>266</td>
<td>TK</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32SLEEP</td>
<td>267</td>
<td>TK</td>
<td>PRE, API</td>
</tr>
</tbody>
</table>
DOS32DEVCONFIG                    268    FS     PRE,API
DOS32GETDATETIME                  269    FS     PRE,API
DOS32SETDATETIME                  270    TIM    PRE,API
DOS32EXECPGM                      271    TIM    PRE,API
DOS32ENTERCRITSEC                 272    TK     PRE,API
DOS32EXITCRITSEC                  273    TK     PRE,API
DOS32EXIT                        274    TK     PRE,API
DOS32KILLPROCESS                  275    TK     PRE,API
DOS32SETPRIORITY                  276    TK     PRE,API
DOS32RESUMETHREAD                 277    TK     PRE,API
DOS32SUSPENDTHREAD                278    TK     PRE,API
DOS32CREATEPIPE                   279    PIP    PRE,API
DOS32CALLNPIPE                    287    PIP    PRE,API
DOS32CONNECTNPIPE                 288    PIP    PRE,API
DOS32DISCONNECTNPIPE              289    PIP    PRE,API
DOS32CREATENPIPE                  290    PIP    PRE,API
DOS32PEEKNPIPE                    291    PIP    PRE,API
DOS32QUERYNPHSTATE                292    PIP    PRE,API
DOS32RAWRREADNPIPE                293    PIP    PRE,API
DOS32RAWWRITENPIPE                294    PIP    PRE,API
DOS32QUERYNPIPEINFO               295    PIP    PRE,API
DOS32QUERYNPIPESEMSSTATE          296    PIP    PRE,API
DOS32SETNPHSTATE                  297    PIP    PRE,API
DOS32SETNPIPESEMSMSTATE           298    PIP    PRE,API
DOS32TRANSACTNPIPE                299    PIP    PRE,API
DOS32WAITNPIPE                    300    PIP    PRE,API
DOS32RESETBUFFER                  301    FS     PRE,API
DOS32SETCURRENTDIR                302    FS     PRE,API
DOS32SETFILEPTR                   303    FS     PRE,API
DOS32PROTECTSETFILEPTR            304    FS     PRE,API
DOS32CLOSE                        305    FS     PRE,API
DOS32PROTECTCLOSE                 306    FS     PRE,API
DOS32COPY                         307    FS     PRE,API
DOS32DELETE                       308    FS     PRE,API
DOS32FORCEDELETE                  309    FS     PRE,API
DOS32DEVI/OCTL                    310    FS     PRE,API
DOS32DUPHANDLE                    311    FS     PRE,API
DOS32EDITNAME                     312    FS     PRE,API
DOS32FINDCLOSE                    313    FS     PRE,API
DOS32FSATTACH 314 FS PRE,API
DOS32FSCTL 315 FS PRE,API
DOS32MOVE 316 FS PRE,API
DOS32SETFILESIZE 317 FS PRE,API
DOS32PROTECTSETFILESIZE 318 FS PRE,API
DOS32QUERYCURRENTDIR 319 FS PRE,API
DOS32QUERYCURRENTDISK 320 FS PRE,API
DOS32QUERYFHVSTATE 321 FS PRE,API
DOS32PROTECTQUERYFHVSTATE 322 FS PRE,API
DOS32PQUERYFSATTACH 323 FS PRE,API
DOS32QUERYFSINFO 324 FS PRE,API
DOS32QUERYHTYPE 325 FS PRE,API
DOS32QUERYVERIFY 326 FS PRE,API
DOS32DELETEDIR 327 FS PRE,API
DOS32SEARCHPATH 328 FS PRE,API
DOS32SETDEFAULTDISK 329 FS PRE,API
DOS32SETFHVSTATE 330 FS PRE,API
DOS32PROTECTSETFHVSTATE 331 FS PRE,API
DOS32SETFSINFO 332 FS PRE,API
DOS32SETMAXFHV 333 FS PRE,API
DOS32SETRELMAXFHV 334 FS PRE,API
DOS32SETVERIFY 335 FS PRE,API
DOS32ERRCLASS 336 FS PRE,API
DOS32ERROR 337 TK PRE,API
DOS32LOADMODULE 338 LDR PRE,API
DOS32FREEMODULE 339 LDR PRE,API
DOS32QUERYMODULEHANDLE 340 LDR PRE,API
DOS32QUERYMODULENAME 341 LDR PRE,API
DOS32QUERYAPPTYPE 342 LDR PRE,API
DOS32PFINDNEXT 343 FS PRE,API
DOS32SHUTDOWN 344 FS PRE,API
DOS32OPENCHANGENOTIFY 345 FS PRE,API
DOS32RESETCHANGENOTIFY 346 FS PRE,API
DOS32CLOSECHANGENOTIFY 347 FS PRE,API
DOS32CREATESPINLOCK 348 LOCK PRE,API
DOS32ACQUIRESPINLOCK 349 LOCK PRE,API
DOS32RELEASESPINLOCK 350 LOCK PRE,API
DOS32FREESPINLOCK 351 LOCK PRE,API
QUECALLS API Tracepoints Indirected Via DOSCALL1

The following table lists pre-invocation tracepoints for QUECALLS.DLL APIs that are indirected via DOSCALL1. These should be trace in conjunction with their corresponding QUECALLS pre-invocation tracepoint.

<table>
<thead>
<tr>
<th>DOSCALL1 API</th>
<th>Minor code</th>
<th>Group</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS32CREATEQUEUE</td>
<td>280</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32OPENQUEUE</td>
<td>281</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32CLOSEQUEUE</td>
<td>282</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32PEEKQUEUE</td>
<td>283</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32PURGEQUEUE</td>
<td>284</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32QUERYQUEUE</td>
<td>285</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32WRITEQUEUE</td>
<td>286</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32READQUEUE</td>
<td>352</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16CREATEQUEUE</td>
<td>353</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16OPENQUEUE</td>
<td>354</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16CLOSEQUEUE</td>
<td>355</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16PEEKQUEUE</td>
<td>356</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16PURGEQUEUE</td>
<td>357</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16QUERYQUEUE</td>
<td>358</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16WRITEQUEUE</td>
<td>359</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS16READQUEUE</td>
<td>360</td>
<td>QUE</td>
<td>PRE, API</td>
</tr>
</tbody>
</table>

MONCALLS.DLL Trace Events

The tracepoints for the MONCALLS.DLL services major code are identified in the following tables. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for MONCALLS Major Code: 0X0010, sorted by minor code.
Trace events for MONCALLS Major Code: 0X0010, sorted by tracepoint.

Trace Events for MONCALLS Major Code: 0X0010, Sorted by
Minor Code

00513 (0X0201) (OS) DosMonClose Pre_Invocation
00514 (0X0202) (OS) DosMonOpen Pre_Invocation
00515 (0X0203) (OS) DosMonRead Pre_Invocation
00516 (0X0204) (OS) DosMonReg Pre_Invocation
00517 (0X0205) (OS) DosMonWrite Pre_Invocation
33281 (0X8201) (OS) DosMonClose Post_Invocation
33282 (0X8202) (OS) DosMonOpen Post_Invocation
33283 (0X8203) (OS) DosMonRead Post_Invocation
33284 (0X8204) (OS) DosMonReg Post_Invocation
33285 (0X8205) (OS) DosMonWrite Post_Invocation

Trace Events for MONCALLS Major Code: 0X0010, Sorted by Tracepoint

(OS) DosMonClose Post_Invocation 33281 (0X8201)
(OS) DosMonClose Pre_Invocation 00513 (0X0201)
(OS) DosMonOpen Post_Invocation 33282 (0X8202)
(OS) DosMonOpen Pre_Invocation 00514 (0X0202)
(OS) DosMonRead Post_Invocation 33283 (0X8203)
(OS) DosMonRead Pre_Invocation 00515 (0X0203)
(OS) DosMonReg Post_Invocation 33284 (0X8204)
(OS) DosMonReg Pre_Invocation 00516 (0X0204)
(OS) DosMonWrite Post_Invocation 33285 (0X8205)
(OS) DosMonWrite Pre_Invocation 00517 (0X0205)

MONCALLS Major Code: 0X0010 Minor Code: 513 (0X0201)

Description
(OS) DosMonClose Pre_Invocation

Tracepoint
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

Minor Code
513 (0X0201)

Trace Groups
TSK

Trace Types
PRE

Traced Parameters
Handle = %w

MONCALLS Major Code: 0X0010 Minor Code: 514 (0X0202)
**Description**
(OS) DosMonOpen Pre_Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**
514 (0X0202)

**Trace Groups**
TSK

**Trace Types**
PRE

**Traced Parameters**
Device name: %s

-------------------------------

**MONCALLS Major Code: 0X0010 Minor Code: 515 (0X0203)**

**Description**
(OS) DosMonRead Pre_Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @monio.c in MONCALLS.

**Minor Code**
515 (0X0203)

**Trace Groups**
TSK

**Trace Types**
PRE

**Traced Parameters**
Input buffer address = %a Wait flag = %w
Data buffer address = %a Byte count = %w

-------------------------------

**MONCALLS Major Code: 0X0010 Minor Code: 516 (0X0204)**

**Description**
(OS) DosMonReg Pre_Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**
516 (0X0204)
**MONCALLS Major Code: 0X0010 Minor Code: 517 (0X0205)**

**Description**  
(OS) DosMonWrite Pre_Invocation

**Tracepoint**  
Source line defined dynamic tracepoint: @monio.c in MONCALLS.

**Minor Code**  
517 (0X0205)

**Trace Groups**  
TSK

**Trace Types**  
PRE

**Traced Parameters**  
Output buffer address = %a Data buffer address = %a  
Byte count = %w Data = %s %b

--------------------------------------------

**MONCALLS Major Code: 0X0010 Minor Code: 33281 (0X8201)**

**Description**  
(OS) DosMonClose Post_Invocation

**Tracepoint**  
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**  
33281 (0X8201)

**Trace Groups**  
TSK

**Trace Types**  
PRE

**Traced Parameters**  
Output buffer address = %a Data buffer address = %a  
Byte count = %w Data = %s %b
MONCALLS Major Code: 0X0010 Minor Code: 33282 (0X8202)

Description
(OS) DosMonOpen Post_Invocation

Tracepoint
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

Minor Code
33282 (0X8202)

Trace Groups
TSK

Trace Types
API

Traced Parameters

Return code = %w New handle = %w

--------------------------------------------

MONCALLS Major Code: 0X0010 Minor Code: 33283 (0X8203)

Description
(OS) DosMonRead Post_Invocation

Tracepoint
Source line defined dynamic tracepoint: @monio.c in MONCALLS.

Minor Code
33283 (0X8203)

Trace Groups
TSK

Trace Types
POST

Traced Parameters

Return code = %w Byte count = %w Data = %r %b

--------------------------------------------
### MONCALLS Major Code: 0X0010 Minor Code: 33284 (0X8204)

**Description**  
(OS) DosMonReg Post Invocation

**Tracepoint**  
Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**  
33284 (0X8204)

**Trace Groups**  
TSK

**Trace Types**  
POST

**Traced Parameters**

- Return code = %w
- Input buffer length = %w Device driver length = %w
- Output buffer length = %w Device driver length = %w

--------------------------------------------

### MONCALLS Major Code: 0X0010 Minor Code: 33285 (0X8205)

**Description**  
(OS) DosMonWrite Post Invocation

**Tracepoint**  
Source line defined dynamic tracepoint: @monio.c in MONCALLS.

**Minor Code**  
33285 (0X8205)

**Trace Groups**  
TSK

**Trace Types**  
POST

**Traced Parameters**

- Return code = %w

--------------------------------------------

### OS2CHAR.DLL Trace Events

The tracepoints for the OS2CHAR.DLL major code are identified in the following tables. These tracepoints are dynamic tracepoints.
Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for OS2CHAR Major Code: 0X0018, sorted by minor code.
Trace events for OS2CHAR Major Code: 0X0018, sorted by tracepoint.

<table>
<thead>
<tr>
<th>Event Code</th>
<th>Minor Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>0X0001</td>
<td>VioAssociate Pre-Invocation</td>
</tr>
<tr>
<td>00002</td>
<td>0X0002</td>
<td>VioCreateLogFont Pre-Invocation</td>
</tr>
<tr>
<td>00003</td>
<td>0X0003</td>
<td>VioCreatePS Pre-Invocation</td>
</tr>
<tr>
<td>00004</td>
<td>0X0004</td>
<td>VioDeleteSetID Pre-Invocation</td>
</tr>
<tr>
<td>00005</td>
<td>0X0005</td>
<td>VioDeRegister Pre-Invocation</td>
</tr>
<tr>
<td>00006</td>
<td>0X0006</td>
<td>VioDestroyPS Pre-Invocation</td>
</tr>
<tr>
<td>00007</td>
<td>0X0007</td>
<td>VioEndPopup Pre-Invocation</td>
</tr>
<tr>
<td>00008</td>
<td>0X0008</td>
<td>VioGetAnsi Pre-Invocation</td>
</tr>
<tr>
<td>00009</td>
<td>0X0009</td>
<td>VioGetBuf Pre-Invocation</td>
</tr>
<tr>
<td>00010</td>
<td>0X000A</td>
<td>VioGetConfig Pre-Invocation</td>
</tr>
<tr>
<td>00011</td>
<td>0X000B</td>
<td>VioGetCp Pre-Invocation</td>
</tr>
<tr>
<td>00012</td>
<td>0X000C</td>
<td>VioGetCurPos Pre-Invocation</td>
</tr>
<tr>
<td>00013</td>
<td>0X000D</td>
<td>VioGetCurType Pre-Invocation</td>
</tr>
<tr>
<td>00014</td>
<td>0X000E</td>
<td>VioGetDeviceCellSize Pre-Invocation</td>
</tr>
<tr>
<td>00015</td>
<td>0X000F</td>
<td>VioGetFont Pre-Invocation</td>
</tr>
<tr>
<td>00016</td>
<td>0X0010</td>
<td>VioGetMode Pre-Invocation</td>
</tr>
<tr>
<td>00017</td>
<td>0X0011</td>
<td>VioGetOrg Pre-Invocation</td>
</tr>
<tr>
<td>00018</td>
<td>0X0012</td>
<td>VioGetPhysBuf Pre-Invocation</td>
</tr>
<tr>
<td>00019</td>
<td>0X0013</td>
<td>VioGetPSAddress Pre-Invocation</td>
</tr>
<tr>
<td>00020</td>
<td>0X0014</td>
<td>VioGetState Pre-Invocation</td>
</tr>
<tr>
<td>00021</td>
<td>0X0015</td>
<td>VioGlobalReg Pre-Invocation</td>
</tr>
<tr>
<td>00022</td>
<td>0X0016</td>
<td>VioModeUndo Pre-Invocation</td>
</tr>
<tr>
<td>00023</td>
<td>0X0017</td>
<td>VioModeWait Pre-Invocation</td>
</tr>
<tr>
<td>00024</td>
<td>0X0018</td>
<td>VioPopUp Pre-Invocation</td>
</tr>
<tr>
<td>00025</td>
<td>0X0019</td>
<td>VioPrintSc Pre-Invocation</td>
</tr>
<tr>
<td>00026</td>
<td>0X001A</td>
<td>VioPrintToggle Pre-Invocation</td>
</tr>
<tr>
<td>00027</td>
<td>0X001B</td>
<td>VioQueryConsole Pre-Invocation</td>
</tr>
<tr>
<td>00028</td>
<td>0X001C</td>
<td>VioQueryFonts Pre-Invocation</td>
</tr>
<tr>
<td>00029</td>
<td>0X001D</td>
<td>VioQuerySetIDs Pre-Invocation</td>
</tr>
<tr>
<td>00030</td>
<td>0X001E</td>
<td>VioReadCellStr Pre-Invocation</td>
</tr>
<tr>
<td>00031</td>
<td>0X001F</td>
<td>VioReadCharStr Pre-Invocation</td>
</tr>
<tr>
<td>00032</td>
<td>0X0020</td>
<td>VioRegister Pre-Invocation</td>
</tr>
<tr>
<td>00033</td>
<td>0X0021</td>
<td>VioSavRedrawUndo Pre-Invocation</td>
</tr>
<tr>
<td>00034</td>
<td>0X0022</td>
<td>VioSavRedrawWait Pre-Invocation</td>
</tr>
<tr>
<td>00035</td>
<td>0X0023</td>
<td>VioScrLock Pre-Invocation</td>
</tr>
<tr>
<td>00036</td>
<td>0X0024</td>
<td>VioScrollDn Pre-Invocation</td>
</tr>
<tr>
<td>00037</td>
<td>0X0025</td>
<td>VioScrollLU Pre-Invocation</td>
</tr>
<tr>
<td>00038</td>
<td>0X0026</td>
<td>VioScrollRT Pre-Invocation</td>
</tr>
<tr>
<td>00039</td>
<td>0X0027</td>
<td>VioScrollUp Pre-Invocation</td>
</tr>
<tr>
<td>00040</td>
<td>0X0028</td>
<td>VioScrUnLock Pre-Invocation</td>
</tr>
<tr>
<td>00041</td>
<td>0X0029</td>
<td>VioSetAnsi Pre-Invocation</td>
</tr>
<tr>
<td>00042</td>
<td>0X002A</td>
<td>VioSetCp Pre-Invocation</td>
</tr>
<tr>
<td>00043</td>
<td>0X002B</td>
<td>VioSetCurPos Pre-Invocation</td>
</tr>
<tr>
<td>00044</td>
<td>0X002C</td>
<td>VioSetCurType Pre-Invocation</td>
</tr>
<tr>
<td>00045</td>
<td>0X002D</td>
<td>VioSetDeviceCellSize Pre-Invocation</td>
</tr>
<tr>
<td>00046</td>
<td>0X002E</td>
<td>VioSetFont Pre-Invocation</td>
</tr>
<tr>
<td>00047</td>
<td>0X002F</td>
<td>VioSetMode Pre-Invocation</td>
</tr>
<tr>
<td>00048</td>
<td>0X0030</td>
<td>VioSetOrg Pre-Invocation</td>
</tr>
<tr>
<td>00049</td>
<td>0X0031</td>
<td>VioSetState Pre-Invocation</td>
</tr>
<tr>
<td>00050</td>
<td>0X0032</td>
<td>VioShieldInit Pre-Invocation</td>
</tr>
<tr>
<td>00051</td>
<td>0X0033</td>
<td>VioShieldTerm Pre-Invocation</td>
</tr>
<tr>
<td>00052</td>
<td>0X0034</td>
<td>VioShowBuf Pre-Invocation</td>
</tr>
</tbody>
</table>
00053 (0X0035)  (OS)  VioShowPS Pre-Invocation
00054 (0X0036)  (OS)  VioWrtCellStr Pre-Invocation
00055 (0X0037)  (OS)  VioWrtCharStr Pre-Invocation
00056 (0X0038)  (OS)  VioWrtCharStrAtt Pre-Invocation
00057 (0X0039)  (OS)  VioWrtNAttr Pre-Invocation
00058 (0X003A)  (OS)  VioWrtNCel Post-Invocation
00059 (0X003B)  (OS)  VioWrtNChar Pre-Invocation
00060 (0X003C)  (OS)  VioWrtTTY Pre-Invocation
00257 (0X0101)  KbdCharIn Pre-Invocation
00258 (0X0102)  KbdClose Pre-Invocation
00259 (0X0103)  KbdGetHWID Pre-Invocation
00260 (0X0104)  KbdDeRegister Pre-Invocation
00261 (0X0105)  KbdFlushBuffer Pre-Invocation
00262 (0X0106)  KbdFreeFocus Pre-Invocation
00263 (0X0107)  KbdGetCP Pre-Invocation
00264 (0X0108)  KbdGetFocus Pre-Invocation
00265 (0X0109)  KbdGetStatus Pre-Invocation
00266 (0X010A)  KbdOpen Pre-Invocation
00267 (0X010B)  KbdPeek Pre-Invocation
00268 (0X010C)  KbdRegister Pre-Invocation
00269 (0X010D)  KbdSetCP Pre-Invocation
00270 (0X010E)  KbdSetCustXT Pre-Invocation
00271 (0X010F)  KbdSetFgnd Pre-Invocation
00272 (0X0110)  KbdSetStatus Pre-Invocation
00273 (0X0111)  KbdShellInit Pre-Invocation
00274 (0X0112)  KbdStringIn Pre-Invocation
00275 (0X0113)  KbdSynch Pre-Invocation
00276 (0X0114)  KbdXlate Pre-Invocation
00513 (0X0201)  (MOU)  MouClose Pre-Invocation
00514 (0X0202)  (MOU)  MouDeRegister Pre-Invocation
00515 (0X0203)  (MOU)  MouDrawPtr Pre-Invocation
00516 (0X0204)  (MOU)  MouFlushQue Pre-Invocation
00517 (0X0205)  (MOU)  MouGetDevStatus Pre-Invocation
00518 (0X0206)  (MOU)  MouGetEventMask Pre-Invocation
00519 (0X0207)  (MOU)  MouGetNumButtons Pre-Invocation
00520 (0X0208)  (MOU)  MouGetNumMickey Pre-Invocation
00521 (0X0209)  (MOU)  MouGetNumQueEl Pre-Invocation
00522 (0X020A)  (MOU)  MouGetPtrPos Pre-Invocation
00523 (0X020B)  (MOU)  MouGetPtrShape Pre-Invocation
00524 (0X020C)  (MOU)  MouGetScaleFact Pre-Invocation
00525 (0X020D)  (MOU)  MouGetThreshold Pre-Invocation
00526 (0X020E)  (MOU)  MouInitReal Pre-Invocation
00527 (0X020F)  (MOU)  MouOpen Pre-Invocation
00528 (0X0210)  (MOU)  MouReadEventQue Pre-Invocation
00529 (0X0211)  (MOU)  MouRegister Pre-Invocation
00530 (0X0212)  (MOU)  MouRemovePtr Pre-Invocation
00531 (0X0213)  (MOU)  MouSetDevStatus Pre-Invocation
00532 (0X0214)  (MOU)  MouSetEventMask Pre-Invocation
00533 (0X0215)  (MOU)  MouSetPtrPos Pre-Invocation
00534 (0X0216)  (MOU)  MouSetPtrShape Pre-Invocation
00535 (0X0217)  (MOU)  MouSetScaleFact Pre-Invocation
00536 (0X0218)  (MOU)  MouSetThreshold Pre-Invocation
00537 (0X0219)  (MOU)  MouShellInit Pre-Invocation
00538 (0X021A)  (MOU)  MouSynch Pre-Invocation
32769 (0X8001)  (OS)  VioAssociate Post-Invocation
32770 (0X8002)  (OS)  VioCreateLogFont Post-Invocation
32771 (0X8003)  (OS)  VioCreatePS Post-Invocation
32772 (0X8004)  (OS)  VioDeleteSetID Post-Invocation
32773 (0X8005)  (OS)  VioDeRegister Post-Invocation
32774 (0X8006)  (OS)  VioDestroyPS Post-Invocation
32775 (0X8007)  (OS)  VioEndPopUp Post-Invocation
32776 (0X8008)  (OS)  VioGetAnsi Post-Invocation
32777 (0X8009)  (OS)  VioGetBuf Post-Invocation
32778 (0X800A)  (OS)  VioGetConfig Post-Invocation
32779 (0X800B)  (OS)  VioGetCP Post-Invocation
32780 (0X800C)  (OS)  VioGetCurPos Post-Invocation
32781 (0X800D)  (OS)  VioGetCurType Post-Invocation
32782 (0X800E)  (OS)  VioGetDeviceCellSize Post-Invocation
32783 (0X800F)  (OS)  VioGetFont Post-Invocation
32784 (0X8010)  (OS)  VioGetMode Post-Invocation
32785 (0X8011)  (OS)  VioGetOrg Post-Invocation
32786 (0X8012)  (OS)  VioGetPhysBuf Post-Invocation
32787 (0X8013)  (OS)  VioGetPSAddress Post-Invocation
32788 (0X8014) (OS) VioGetState Post-Invocation
32789 (0X8015) (OS) VioGlobalReg Post-Invocation
32790 (0X8016) (OS) VioModeUndo Post-Invocation
32791 (0X8017) (OS) VioModeWait Post-Invocation
32792 (0X8018) (OS) VioPopUp Post-Invocation
32793 (0X8019) (OS) VioPrtSc Post-Invocation
32794 (0X801A) (OS) VioPrtScToggle Post-Invocation
32795 (0X801B) (OS) VioQueryConsole Post-Invocation
32796 (0X801C) (OS) VioQueryFonts Post-Invocation
32797 (0X801D) (OS) VioQuerySetIDs Post-Invocation
32798 (0X801E) (OS) VioReadCellStr Post-Invocation
32799 (0X801F) (OS) VioReadCharStr Post-Invocation
32800 (0X8020) (OS) VioRegister Post-Invocation
32801 (0X8021) (OS) VioSavRedrawUndo Post-Invocation
32802 (0X8022) (OS) VioSavRedrawWait Post-Invocation
32803 (0X8023) (OS) VioScrLock Post-Invocation
32804 (0X8024) (OS) VioScrollDn Post-Invocation
32805 (0X8025) (OS) VioScrollLf Post-Invocation
32806 (0X8026) (OS) VioScrollRt Post-Invocation
32807 (0X8027) (OS) VioScrollUp Post-Invocation
32808 (0X8028) (OS) VioScrUnLock Post-Invocation
32809 (0X8029) (OS) VioSetAnsi Post-Invocation
32810 (0X802A) (OS) VioSetCp Post-Invocation
32811 (0X802B) (OS) VioSetCurPos Post-Invocation
32812 (0X802C) (OS) VioSetCurType Post-Invocation
32813 (0X802D) (OS) VioSetDeviceCellSize Post-Invocation
32814 (0X802E) (OS) VioSetFont Post-Invocation
32815 (0X802F) (OS) VioSetMode Post-Invocation
32816 (0X8030) (OS) VioSetOrg Post-Invocation
32817 (0X8031) (OS) VioSetState Post-Invocation
32818 (0X8032) (OS) VioShieldInit Post-Invocation
32819 (0X8033) (OS) VioShieldTerm Post-Invocation
32820 (0X8034) (OS) VioShowBuf Post-Invocation
32821 (0X8035) (OS) VioShowPS Post-Invocation
32822 (0X8036) (OS) VioWrtCellStr Post-Invocation
32823 (0X8037) (OS) VioWrtCharStr Post-Invocation
32824 (0X8038) (OS) VioWrtCharStrAtt Post-Invocation
32825 (0X8039) (OS) VioWrtNAttr Post-Invocation
32826 (0X803A) (OS) VioWrtNCell Post-Invocation
32827 (0X803B) (OS) VioWrtTTY Post-Invocation
33025 (0X8101) KbdCharIn Post-Invocation
33026 (0X8102) KbdClose Post-Invocation
33027 (0X8103) KbdGetHWID Post-Invocation
33028 (0X8104) KbdDeRegister Post-Invocation
33029 (0X8105) KbdFlushBuffer Post-Invocation
33030 (0X8106) KbdFreeFocus Post-Invocation
33031 (0X8107) KbdGetCP Post-Invocation
33032 (0X8108) KbdGetFocus Post-Invocation
33033 (0X8109) KbdGetStatus Post-Invocation
33034 (0X810A) KbdOpen Post-Invocation
33035 (0X810B) KbdPeek Post-Invocation
33036 (0X810C) KbdRegister Post-Invocation
33037 (0X810D) KbdSetCP Post-Invocation
33038 (0X810E) KbdSetCustXT Post-Invocation
33039 (0X810F) KbdSetFgnd Post-Invocation
33040 (0X8110) KbdSetStatus Post-Invocation
33041 (0X8111) KbdShellInit Post-Invocation
33042 (0X8112) KbdStringIn Post-Invocation
33043 (0X8113) KbdSynch Post-Invocation
33044 (0X8114) KbdXlate Post-Invocation
33281 (0X8201) (MOU) MouClose Post-Invocation
33282 (0X8202) (MOU) MouDeRegister Post-Invocation
33283 (0X8203) (MOU) MouDrawPtr Post-Invocation
33284 (0X8204) (MOU) MouFlushQue Post-Invocation
33285 (0X8205) (MOU) MouGetDevStatus Post-Invocation
33286 (0X8206) (MOU) MouGetEventMask Post-Invocation
33287 (0X8207) (MOU) MouGetNumButtons Post-Invocation
33288 (0X8208) (MOU) MouGetNumMiceKeys Post-Invocation
33289 (0X8209) (MOU) MouGetNumQueEl Post-Invocation
33290 (0X820A) (MOU) MouGetPtrPos Post-Invocation
33291 (0X820B) (MOU) MouGetPtrShape Post-Invocation
33292 (0X820C) (MOU) MouGetScaleFact Post-Invocation
Trace Events for OS2CHAR Major Code: 0X0018, Sorted by Tracepoint

KBDCHARIN 00257 (0X0101)
KBDCHARIN 33025 (0X8101)
KBDCLOSE 00258 (0X0102)
KBDCLOSE 33026 (0X8102)
KBDDEREGISTER 00260 (0X0104)
KBDDEREGISTER 33028 (0X8104)
KBDFLUSHBUFFER 00261 (0X0105)
KBDFLUSHBUFFER 33029 (0X8105)
KBDFREEFOCUS 00262 (0X0106)
KBDFREEFOCUS 33030 (0X8106)
KBGETTCP 00263 (0X0107)
KBGETTCP 33031 (0X8107)
KBGETFOCUS 00264 (0X0108)
KBGETFOCUS 33032 (0X8108)
KBGETHWID 00259 (0X0103)
KBGETHWID 33027 (0X8103)
KBGETSTATUS 00265 (0X0109)
KBGETSTATUS 33033 (0X8109)
KBOPEN 00266 (0X010A)
KBOPEN 33034 (0X810A)
KBPEEK 00267 (0X010B)
KBPEEK 33035 (0X810B)
KBREGISTER 00268 (0X010C)
KBREGISTER 33036 (0X810C)
KBSETTCP 00269 (0X010D)
KBSETTCP 33037 (0X810D)
KBSETCUSTXT 00270 (0X010E)
KBSETCUSTXT 33038 (0X810E)
KBSETFGND 00271 (0X010F)
KBSETFGND 33039 (0X810F)
KBSETSTATUS 00272 (0X0110)
KBSETSTATUS 33040 (0X8110)
KBSETSTRING 00273 (0X0111)
KBSETSTRING 33041 (0X8111)
KBSTRING 00274 (0X0112)
KBSTRING 33042 (0X8112)
KBSYNCH 00275 (0X0113)
KBSYNCH 33043 (0X8113)
KBDXLADE 00276 (0X0114)
KBDXLADE 33044 (0X8114)
MOUCLOSE 00513 (0X0201)
MOUCLOSE 33281 (0X8201)
MOUDEREGISTER 00514 (0X0202)
MOUDEREGISTER 33282 (0X8202)
MOUDRAWPTR 00515 (0X0203)
MOUDRAWPTR 33283 (0X8203)
OS2CHAR Major Code: 0X0018 Minor Code: 1 (0X0001)

**Description**  
(OS) VioAssociate Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOASSOCIATE

**Minor Code**  
1 (0X0001)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

- PS Handle = %w
- Device Context Handle = %d

OS2CHAR Major Code: 0X0018 Minor Code: 2 (0X0002)

**Description**  
(OS) VioCreateLogFont Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATELOGFONT

**Minor Code**  
2 (0X0002)

**Trace Groups**  
VIO
**Trace Types**

PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Logical Font Name</td>
<td>%s</td>
</tr>
<tr>
<td>Local Identifier</td>
<td>%d</td>
</tr>
<tr>
<td>FATTRRS - usRecordLength</td>
<td>%w</td>
</tr>
<tr>
<td>- fsSelection</td>
<td>%w</td>
</tr>
<tr>
<td>- lMatch</td>
<td>%d</td>
</tr>
<tr>
<td>- szFacename</td>
<td>%s</td>
</tr>
<tr>
<td>- idRegistry</td>
<td>%w</td>
</tr>
<tr>
<td>- usCodePage</td>
<td>%w</td>
</tr>
<tr>
<td>- lMaxBaseLineExt</td>
<td>%d</td>
</tr>
<tr>
<td>- lAveCharWidth</td>
<td>%d</td>
</tr>
<tr>
<td>- fsType</td>
<td>%w</td>
</tr>
<tr>
<td>- fsFontUse</td>
<td>%w</td>
</tr>
</tbody>
</table>

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 3 (0X0003)**

**Description**

(OS) VioCreatePS Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATEPS

**Minor Code**

3 (0X0003)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>%w</td>
</tr>
<tr>
<td>Attribute Size</td>
<td>%w</td>
</tr>
<tr>
<td>Format</td>
<td>%w</td>
</tr>
<tr>
<td>Width</td>
<td>%w</td>
</tr>
<tr>
<td>Depth</td>
<td>%w</td>
</tr>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>
OS2CHAR Major Code: 0X0018 Minor Code: 4 (0X0004)

**Description**
(OS) VioDeleteSetID Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIODELETESETID

**Minor Code**
4 (0X0004)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>PS Handle</th>
<th>%w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Identifier</td>
<td>%d</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 5 (0X0005)

**Description**
(OS) VioDeRegister Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIODEREGISTER

**Minor Code**
5 (0X0005)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

None

OS2CHAR Major Code: 0X0018 Minor Code: 6 (0X0006)

**Description**
(OS) VioDestroyPS Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIODESTROYPS

**Minor Code**
6 (0X0006)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 7 (0X0007)

**Description**
(OS) VioEndPopUp Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOENDPOPUP

**Minor Code**
7 (0X0007)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 8 (0X0008)

**Description**
(OS) VioGetAnsi Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETANSI

**Minor Code**
8 (0X0008)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>
OS2CHAR Major Code: 0X0018 Minor Code: 9 (0X0009)

Description
(OS) VioGetBuf Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETBUF

Minor Code
9 (0X0009)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

Handle = %w
Length = %w
LVBptr = %d

OS2CHAR Major Code: 0X0018 Minor Code: 10 (0X000A)

Description
(OS) VioGetConfig Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCONFIG

Minor Code
10 (0X000A)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

Handle = %w
Config Data = %r%w
Reserved = %w
OS2CHAR Major Code: 0X0018 Minor Code: 11 (0X000B)

Description
(OS) VioGetCp Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCP

Minor Code
11 (0X000B)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
- Handle = %w
- Code Page Id = %w
- Reserved = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 12 (0X000C)

Description
(OS) VioGetCurPos Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURPOS

Minor Code
12 (0X000C)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
- Handle = %w
- Column = %w
- Row = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 13 (0X000D)
**Description**
(OS) VioGetCurType Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURTYPE

**Minor Code**
13 (0X000D)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

Handle = %w
Cursor Data = %w%w%w%w

-------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 14 (0X000E)**

**Description**
(OS) VioGetDeviceCellSize Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETDEVICECELLSIZE

**Minor Code**
14 (0X000E)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

PS Handle = %w
Width = %w
Height = %w

-------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 15 (0X000F)**

**Description**
(OS) VioGetFont Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETFONT

**Minor Code**
15 (0X000F)
Trace Groups  VIO
Trace Types   PRE
Traced Parameters

Handle                   = %w
Request Block           = %r%w

-------------------------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 16 (0X0010)

Description     (OS) VioGetMode Pre-Invocation
Tracepoint      Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETMODE
Minor Code      16 (0X0010)
Trace Groups    VIO
Trace Types     PRE
Traced Parameters

Handle                   = %w
ModeData                 = %r%w

-------------------------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 17 (0X0011)

Description     (OS) VioGetOrg Pre-Invocation
Tracepoint      Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETORG
Minor Code      17 (0X0011)
Trace Groups    VIO
Trace Types     PRE
Traced Parameters
OS2CHAR Major Code: 0X0018 Minor Code: 18 (0X0012)

Description
(OS) VioGetPhysBuf Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPHYSBUF

Minor Code
18 (0X0012)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
Reserved = %w
DisplayBuf = %d%d%w%w

OS2CHAR Major Code: 0X0018 Minor Code: 19 (0X0013)

Description
(OS) VioGetPSAddress Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPSADDRESS

Minor Code
19 (0X0013)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
PS Handle = %w
PS Address = %d
Reserved = %d

OS2CHAR Major Code: 0X0018 Minor Code: 20 (0X0014)
**Description**  
(OS) VioGetState Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETSTATE

**Minor Code**  
20 (0X0014)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Request Block</td>
<td>%r%w</td>
</tr>
</tbody>
</table>

-------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 21 (0X0015)**

**Description**  
(OS) VioGlobalReg Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGLOBALREG

**Minor Code**  
21 (0X0015)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>%d</td>
</tr>
<tr>
<td>Function Mask 2</td>
<td>%d</td>
</tr>
<tr>
<td>Function Mask 1</td>
<td>%d</td>
</tr>
<tr>
<td>Entry Point Name</td>
<td>%s</td>
</tr>
<tr>
<td>Module Name</td>
<td>%s</td>
</tr>
</tbody>
</table>

-------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 22 (0X0016)**

**Description**
(OS) VioModeUndo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEUNDO

Minor Code
22 (0X0016)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

Reserved = \%w
Kill Indicator = \%w
Owner Indicator = \%w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 23 (0X0017)

Description
(OS) VioModeWait Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEWAIT

Minor Code
23 (0X0017)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

Reserved = \%w
Notify Type = \%w
Request Type = \%w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 24 (0X0018)

Description
(OS) VioPopUp Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPopup

Minor Code
24 (0X0018)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Options</td>
<td>%w</td>
</tr>
</tbody>
</table>

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 25 (0X0019)**

**Description**  
(OS) VioPrtSc Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSC

**Minor Code**  
25 (0X0019)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 26 (0X001A)**

**Description**  
(OS) VioPrtScToggle Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSCTOGGLE

**Minor Code**  
26 (0X001A)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
</tbody>
</table>
OS2CHAR Major Code: 0X0018 Minor Code: 27 (0X001B)

Description
(OS) VioQueryConsole Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYCONSOLE

Minor Code
27 (0X001B)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

   Console Structure  = %d

OS2CHAR Major Code: 0X0018 Minor Code: 28 (0X001C)

Description
(OS) VioQueryFonts Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYFONTS

Minor Code
28 (0X001C)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

   PS Handle          = %w
   Options            = %d
   Font Face Name     = %s
   Fonts Returned     = %d
   Metrics Length     = %d

OS2CHAR Major Code: 0X0018 Minor Code: 29 (0X001D)
Description  
(OS) VioQuerySetIDs Pre-Invocation

Tracepoint 
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYSETIDS

Minor Code 
29 (0X001D)

Trace Groups  
VIO

Trace Types  
PRE

Traced Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Count</td>
<td>%d</td>
</tr>
<tr>
<td>Types</td>
<td>%d</td>
</tr>
<tr>
<td>Font Names</td>
<td>%r%b</td>
</tr>
<tr>
<td>Local Identifiers</td>
<td>%d</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 30 (0X001E)

Description  
(OS) VioReadCellStr Pre-Invocation

Tracepoint 
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCELLSTR

Minor Code 
30 (0X001E)

Trace Groups  
VIO

Trace Types  
PRE

Traced Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Column</td>
<td>%w</td>
</tr>
<tr>
<td>Row</td>
<td>%w</td>
</tr>
<tr>
<td>Length</td>
<td>%w</td>
</tr>
<tr>
<td>CellStr</td>
<td>%r%b</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 31 (0X001F)
**Description**  
(OS) VioReadCharStr Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCHARSTR

**Minor Code**  
31 (0X001F)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

- Handle = %w
- Column = %w
- Row = %w
- Length = %w
- CharStr = %s%b

OS2CHAR Major Code: 0X0018 Minor Code: 32 (0X0020)

**Description**  
(OS) VioRegister Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREGISTER

**Minor Code**  
32 (0X0020)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

- Function Mask 2 = %d
- Function Mask 1 = %d
- Entry Point Name = %s
- Module Name = %s

OS2CHAR Major Code: 0X0018 Minor Code: 33 (0X0021)
**Description**  
(OS) VioSavRedrawUndo Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWUNDO

**Minor Code**  
33 (0X0021)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

Reserved = %w  
Kill Indicator = %w  
Owner Indicator = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 34 (0X0022)

**Description**  
(OS) VioSavRedrawWait Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWWAIT

**Minor Code**  
34 (0X0022)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

Reserved = %w  
Notify Type = %w  
SavRedrawIndic = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 35 (0X0023)

**Description**  
(OS) VioScrLock Pre-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minor Code</strong></td>
<td>35 (0X0023)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>VIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Status</td>
<td>%b</td>
</tr>
<tr>
<td>Wait Flag</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 36 (0X0024)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) VioScrollDn Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLDN</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>36 (0X0024)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>VIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Cell</td>
<td>%w</td>
</tr>
<tr>
<td>Lines</td>
<td>%w</td>
</tr>
<tr>
<td>RightCol</td>
<td>%w</td>
</tr>
<tr>
<td>BotRow</td>
<td>%w</td>
</tr>
<tr>
<td>LeftCol</td>
<td>%w</td>
</tr>
<tr>
<td>TopRow</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 37 (0X0025)

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
(OS) VioScrollLf Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLLF

**Minor Code**
37 (0X0025)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

- Handle = %w
- Cell = %w
- Lines = %w
- RightCol = %w
- BotRow = %w
- LeftCol = %w
- TopRow = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 38 (0X0026)

**Description**
(OS) VioScrollRt Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLRT

**Minor Code**
38 (0X0026)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

- Handle = %w
- Cell = %w
- Lines = %w
- RightCol = %w
- BotRow = %w
- LeftCol = %w
- TopRow = %w

--------------------------------------------
OS2CHAR Major Code: 0X0018 Minor Code: 39 (0X0027)

**Description**  
(OS) VioScrollUp Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLUP

**Minor Code**  
39 (0X0027)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

- Handle = %w
- Cell = %w
- Lines = %w
- RightCol = %w
- BotRow = %w
- LeftCol = %w
- TopRow = %w

OS2CHAR Major Code: 0X0018 Minor Code: 40 (0X0028)

**Description**  
(OS) VioScrUnLock Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLUP

**Minor Code**  
40 (0X0028)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

- Handle = %w
OS2CHAR Major Code: 0X0018 Minor Code: 41 (0X0029)

Description  (OS) VioSetAnsi Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETANSI

Minor Code  41 (0X0029)

Trace Groups  VIO

Trace Types  PRE

Traced Parameters

Handle                   = %w
Indicator               = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 42 (0X002A)

Description  (OS) VioSetCp Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETTCP

Minor Code  42 (0X002A)

Trace Groups  VIO

Trace Types  PRE

Traced Parameters

Handle                   = %w
Code Page Id             = %w
Reserved                 = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 43 (0X002B)

Description  (OS) VioSetCurPos Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURPOS

Minor Code
43 (0X002B)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
Handle = %w
Column = %w
Row = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 44 (0X002C)

Description
(OS) VioSetCurType Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURTYPE

Minor Code
44 (0X002C)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters
Handle = %w
CursorData = %w%w%w%w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 45 (0X002D)

Description
(OS) VioSetDeviceCellSize Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETDEVICECELLSIZE

Minor Code
45 (0X002D)

Trace Groups
OS2CHAR Major Code: 0X0018 Minor Code: 46 (0X002E)

**Description**
(OS) VioSetFont Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETFONT

**Minor Code**
46 (0X002E)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**
Handle = %w
Request Block = %r%w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 47 (0X002F)

**Description**
(OS) VioSetMode Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETMODE

**Minor Code**
47 (0X002F)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**
OS2CHAR Major Code: 0X0018 Minor Code: 48 (0X0030)

**Description**
(OS) VioSetOrg Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETORG

**Minor Code**
48 (0X0030)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

Handle = %w
Column = %w
Row = %w

OS2CHAR Major Code: 0X0018 Minor Code: 49 (0X0031)

**Description**
(OS) VioSetState Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETSTATE

**Minor Code**
49 (0X0031)

**Trace Groups**
VIO

**Trace Types**
PRE

**Traced Parameters**

Handle = %w
Request Block = %r%w

OS2CHAR Major Code: 0X0018 Minor Code: 50 (0X0032)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) VioShieldInit Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDINIT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>50 (0X0032)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>VIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>None</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 51 (0X0033)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) VioShieldTerm Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDTERM</td>
</tr>
<tr>
<td>Minor Code</td>
<td>51 (0X0033)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>VIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>None</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 52 (0X0034)

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) VioShowBuf Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWBUF</td>
</tr>
<tr>
<td>Minor Code</td>
<td>52 (0X0034)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>VIO</td>
</tr>
</tbody>
</table>
Trace Types
PRE

Traced Parameters

Handle = %w
Length = %w
Offset = %w

OS2CHAR Major Code: 0X0018 Minor Code: 53 (0X0035)

Description
(OS) VioShowPS Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWPS

Minor Code
53 (0X0035)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

PS Handle = %w
Cell Offset = %w
Width = %w
Depth = %w

OS2CHAR Major Code: 0X0018 Minor Code: 54 (0X0036)

Description
(OS) VioWrtCellStr Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCELLSTR

Minor Code
54 (0X0036)

Trace Groups
VIO

Trace Types
PRE
Traced Parameters

Handle = %w
Column = %w
Row = %w
Length = %w
Cell String = %r%b

OS2CHAR Major Code: 0X0018 Minor Code: 55 (0X0037)

Description
(OS) VioWrtCharStr Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTR

Minor Code
55 (0X0037)

Trace Groups
VIO

Trace Types
PRE

Traced Parameters

Handle = %w
Column = %w
Row = %w
Length = %w
Char String = %r%b

OS2CHAR Major Code: 0X0018 Minor Code: 56 (0X0038)

Description
(OS) VioWrtCharStrAtt Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTRATT

Minor Code
56 (0X0038)

Trace Groups
VIO

Trace Types
**Traced Parameters**

Handle = %w
Attr = %b
Column = %w
Row = %w
Length = %w
CharStr = %r%b

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 57 (0X0039)**

**Description**  
(OS) VioWrtNAttr Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNATTR

**Minor Code**  
57 (0X0039)

**Trace Groups**  
VIO

**Trace Types**  
PRE

**Traced Parameters**

Handle = %w
Column = %w
Row = %w
Times = %w
Attr = %b

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 58 (0X003A)**

**Description**  
(OS) VioWrtNCell Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCCELL

**Minor Code**  
58 (0X003A)
Trace Groups: VIO
Trace Types: PRE
Traced Parameters:
Handle = %w
Column = %w
Row = %w
Times = %w
Char = %b

OS2CHAR Major Code: 0X0018 Minor Code: 59 (0X003B)

Description: (OS) VioWrtNChar Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCHAR
Minor Code: 59 (0X003B)
Trace Groups: VIO
Trace Types: PRE
Traced Parameters:
Handle = %w
Column = %w
Row = %w
Times = %w
Char = %b

OS2CHAR Major Code: 0X0018 Minor Code: 60 (0X003C)

Description: (OS) VioWrtTTY Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTTTY
Minor Code
OS2CHAR Major Code: 0X0018 Minor Code: 257 (0X0101)

Description
KbdCharIn Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDCHARIN

Minor Code
257 (0X0101)

Trace Groups
KBD

Trace Types
PRE

Traced Parameters

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
IOWAIT = %W
CharDataRec addr. = %A

OS2CHAR Major Code: 0X0018 Minor Code: 258 (0X0102)

Description
KbdClose Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDCLOSE
Minor Code
258 (0X0102)

Trace Groups
KBD

Trace Types
PRE

Traced Parameters

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W

--------------------------------------------------------------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 259 (0X0103)

Description
KbdGetHWID Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETHWID

Minor Code
259 (0X0103)

Trace Groups
KBD

Trace Types
PRE

Traced Parameters

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
HWIDStruc addr. = %A
HWIDStruc length = %W

--------------------------------------------------------------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 260 (0X0104)
Description: KbdDeRegister Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDDEREGISTER

Minor Code: 260 (0X0104)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W

OS2CHAR Major Code: 0X0018 Minor Code: 261 (0X0105)

Description: KbdFlushBuffer Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDFLUSHBUFFER

Minor Code: 261 (0X0105)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W

OS2CHAR Major Code: 0X0018 Minor Code: 262 (0X0106)
**Description**
KbdFreeFocus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDFREEFOCUS

**Minor Code**
262 (0X0106)

**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- **Major** = %X
- **Minor** = %Y
- **ProcStatus** = %B
- **ProcType** = %B
- **SessionID** = %W
- **KbdHandle** = %W

-------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 263 (0X0107)**

**Description**
KbdGetCP Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETCP

**Minor Code**
263 (0X0107)

**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- **Major** = %X
- **Minor** = %Y
- **ProcStatus** = %B
- **ProcType** = %B
- **SessionID** = %W
- **KbdHandle** = %W
- **CodePageID addr.** = %A

-------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 264 (0X0108)**
Description: KbdGetFocus Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETFOCUS

Minor Code: 264 (0X0108)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

Major = %X Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
IOWAIT = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 265 (0X0109)

Description: KbdGetStatus Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETSTATUS

Minor Code: 265 (0X0109)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

Major = %X Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
Status addr. = %A
OS2CHAR Major Code: 0X0018 Minor Code: 266 (0X010A)

Description: KbdOpen Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDOPEN
Minor Code: 266 (0X010A)
Trace Groups: KBD
Trace Types: PRE
Traced Parameters:

- Major = %X  Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle addr. = %A

OS2CHAR Major Code: 0X0018 Minor Code: 267 (0X010B)

Description: KbdPeek Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDPEEK
Minor Code: 267 (0X010B)
Trace Groups: KBD
Trace Types: PRE
Traced Parameters:

- Major = %X  Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W
OS2CHAR Major Code: 0X0018 Minor Code: 268 (0X010C)

Description
KbdRegister Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDREGISTER

Minor Code
268 (0X010C)

Trace Groups
KBD

Trace Types
PRE

Traced Parameters

Major = %X  Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
Function Mask = %D
EntryName addr. = %A
EntryName = %S
ModuleName addr. = %A
ModuleName = %S

OS2CHAR Major Code: 0X0018 Minor Code: 269 (0X010D)

Description
KbdSetCP Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCP

Minor Code
269 (0X010D)

Trace Groups
KBD

Trace Types
PRE
Traced Parameters

Major = %X   Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
CodePageID = %W

OS2CHAR Major Code: 0X0018 Minor Code: 270 (0X010E)

Description
KbdSetCustXT Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCUSTXT

Minor Code
270 (0X010E)

Trace Groups
KBD

Trace Types
PRE

Traced Parameters

Major = %X   Minor = %Y
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
XlateTable addr. = %A

OS2CHAR Major Code: 0X0018 Minor Code: 271 (0X010F)

Description
KbdSetFgnd Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETFGND

Minor Code
271 (0X010F)
**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 272 (0X0110)**

**Description**
KbdSetStatus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETSTATUS

**Minor Code**
272 (0X0110)

**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W
- Status addr. = %A
- Status = %W
- %W
- %W
- %W
- %W

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 273 (0X0111)**
Description: KbdShellInit Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDSHELLINIT

Minor Code: 273 (0X0111)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 274 (0X0112)

Description: KbdStringIn Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDSTRINGIN

Minor Code: 274 (0X0112)

Trace Groups: KBD

Trace Types: PRE

Traced Parameters:

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W
- IOWAIT = %W
- LengthBuffer addr. = %A
- CharBuffer addr. = %A
OS2CHAR Major Code: 0X0018 Minor Code: 275 (0X0113)

**Description**
KbdSynch Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSYNCH

**Minor Code**
275 (0X0113)

**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- IOWAIT = %W

OS2CHAR Major Code: 0X0018 Minor Code: 276 (0X0114)

**Description**
KbdXlate Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDXLATE

**Minor Code**
276 (0X0114)

**Trace Groups**
KBD

**Trace Types**
PRE

**Traced Parameters**

- Major = %X
- Minor = %Y
- ProcStatus = %B
- ProcType = %B
SessionID = %W
KbdHandle  = %W
XlateRecord addr. = %A
XlateRecord    = %B %B %B %B
 %B %B %B %B
 %B %B %B %B
 %B %B %B %B
 %B %B %B %B
 %B %B

--------------------------------------------

OS2CHAR Major Code: 0X0018Minor Code: 513 (0X0201)

Description
(MOU) MouClose Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUCLOSE

Minor Code
513 (0X0201)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters
ProcStatus = %b
ProcType   = %b
SessionID = %w
MouHandle  = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018Minor Code: 514 (0X0202)

Description
(MOU) MouDeRegister Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUDEREGISTER

Minor Code
514 (0X0202)

Trace Groups
MOU
**Trace Types**  
PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 515 (0X0203)**

**Description**  
(MOU) MouDrawPtr Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.MOUDRAWPTR

**Minor Code**  
515 (0X0203)

**Trace Groups**  
MOU

**Trace Types**  
PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 516 (0X0204)**

**Description**  
(MOU) MouFlushQue Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.MOUFLUSHQUE

**Minor Code**  
516 (0X0204)

**Trace Groups**  
MOU

**Trace Types**  
PRE

**Traced Parameters**
OS2CHAR Major Code: 0X0018 Minor Code: 517 (0X0205)

**Description**
(MOU) MouGetDevStatus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETDEVSTATUS

**Minor Code**
517 (0X0205)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**

- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w
- DevStatus addr. = %a

--------------------------------------------
OS2CHAR Major Code: 0X0018 Minor Code: 518 (0X0206)

**Description**
(MOU) MouGetEventMask Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETEVENTMASK

**Minor Code**
518 (0X0206)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**
OS2CHAR Major Code: 0X0018 Minor Code: 519 (0X0207)

Description:  (MOU) MouGetNumButtons Pre-Invocation

Tracepoint:  Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMBUTTONS

Minor Code:  519 (0X0207)

Trace Groups:  MOU

Trace Types:  PRE

Traced Parameters:

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
NumButtons addr. = %a

OS2CHAR Major Code: 0X0018 Minor Code: 520 (0X0208)

Description:  (MOU) MouGetNumMickeys Pre-Invocation

Tracepoint:  Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMMICKEYS

Minor Code:  520 (0X0208)

Trace Groups:  MOU

Trace Types:  PRE
Traced Parameters

ProcStatus = %b
ProcType   = %b
SessionID = %w
MouHandle = %w
NumMickey's addr. = %a

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 521 (0X0209)

Description
(MOU) MouGetNumQueEl Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMQUEEL

Minor Code
521 (0X0209)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType   = %b
SessionID = %w
MouHandle = %w
QueDataRec addr. = %a

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 522 (0X020A)

Description
(MOU) MouGetPtrPos Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRPOS

Minor Code
522 (0X020A)

Trace Groups
MOU
Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
PtrPos addr. = %a

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 523 (0X020B)

Description
(MOU) MouGetPtrShape Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRSHAPE

Minor Code
523 (0X020B)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
PtrDefRec addr. = %a
PtrBuffer addr. = %a
PtrBuffer length = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 524 (0X020C)

Description
(MOU) MouGetScaleFact Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETSCALEFACT
Minor Code 524 (0X020C)
Trace Groups MOU
Trace Types PRE
Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
ScaleStruc addr. = %a

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 525 (0X020D)

Description (MOU) MouGetThreshold Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETTHRESHOLD
Minor Code 525 (0X020D)
Trace Groups MOU
Trace Types PRE
Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
ThresholdStruc addr. = %a
ThresholdStruc Length = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 526 (0X020E)

Description (MOU) MouInitReal Pre-Invocation
OS2CHAR Major Code: 0X0018 Minor Code: 526 (0X020E)

Description
(MOU) MouInitReal Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUINITREAL

Minor Code
526 (0X020E)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w

OS2CHAR Major Code: 0X0018 Minor Code: 527 (0X020F)

Description
(MOU) MouOpen Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUOPEN

Minor Code
527 (0X020F)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle addr. = %a
PtrName addr. = %a
PtrName = %s

OS2CHAR Major Code: 0X0018 Minor Code: 528 (0X0210)

Description
(MOU) MouReadEventQue Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUREADEVENTQUE

Minor Code
528 (0X0210)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
EventBuf addr. = %a
ReadType addr. = %a
ReadType = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 529 (0X0211)

Description
(MOU) MouRegister Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUREGISTER

Minor Code
529 (0X0211)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
Function Mask = %d
EntryName addr. = %a
EntryName = %s
ModuleName addr. = %a
ModuleName = %s
OS2CHAR Major Code: 0X0018 Minor Code: 530 (0X0212)

**Description**
(MOU) MouRemovePtr Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUREMOVEPTR

**Minor Code**
530 (0X0212)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**
- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w
- PrtRec addr. = %a
- PrtRec.UpLEFTROW = %w
- .UpLEFTCOL = %w
- .LoRIGHTROW = %w
- .LoRIGHTCOL = %w

OS2CHAR Major Code: 0X0018 Minor Code: 531 (0X0213)

**Description**
(MOU) MouSetDevStatus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETDEVSTATUS

**Minor Code**
531 (0X0213)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**
OS2CHAR Major Code: 0X0018 Minor Code: 532 (0X0214)

Description
(MOU) MouSetEventMask Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETEVENTMASK

Minor Code
532 (0X0214)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters

ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
DevStatus addr. = %a
DevStatus = %w

OS2CHAR Major Code: 0X0018 Minor Code: 533 (0X0215)

Description
(MOU) MouSetPtrPos Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETPTRPOS

Minor Code
533 (0X0215)

Trace Groups
MOU
Trace Types
PRE

Traced Parameters
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
PtrPos addr. = %a
PtrPos.ROW = %w
.COL = %w

OS2CHAR Major Code: 0X0018 Minor Code: 534 (0X0216)

Description
(MOU) MouSetPtrShape Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETPTRSHAPE

Minor Code
534 (0X0216)

Trace Groups
MOU

Trace Types
PRE

Traced Parameters
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
PtrBuffer addr. = %a
PtrDefRec addr. = %a
PtrDefRec.BUFFERLEN = %w
.COL = %w
.ROW = %w
.COLOFFSET = %w
.ROWOFFSET = %w

---------------------------------------------
OS2CHAR Major Code: 0X0018 Minor Code: 535 (0X0217)

**Description**
(MOU) MouSetScaleFact Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETSCALEFACT

**Minor Code**
535 (0X0217)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**

- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w
- ScaleFact addr. = %a
- ScaleFact.ROWSCALE = %w
- .COLSCALE = %w

OS2CHAR Major Code: 0X0018 Minor Code: 536 (0X0218)

**Description**
(MOU) MouSetThreshold Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETTHRESHOLD

**Minor Code**
536 (0X0218)

**Trace Groups**
MOU

**Trace Types**
PRE

**Traced Parameters**

- ProcStatus = %b
- ProcType = %b
- SessionID = %w
MouHandle = %w
ThresholdStruc addr. = %a
ThresholdStruc.Length     = %w
.Level1     = %w
.Lev1Mult   = %w
.Level2     = %w
.Lev2Mult   = %w

OS2CHAR Major Code: 0X0018 Minor Code: 537 (0X0219)

Description (MOU) MouShellInit Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: OS2CHAR.MOUSHELLINIT
Minor Code 537 (0X0219)
Trace Groups MOU
Trace Types PRE
Traced Parameters
ProcStatus = %b
ProcType   = %b
SessionID = %w

OS2CHAR Major Code: 0X0018 Minor Code: 538 (0X021A)

Description (MOU) MouSynch Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: OS2CHAR.MOUSYNCH
Minor Code 538 (0X021A)
Trace Groups MOU
Trace Types PRE
Traced Parameters
OS2CHAR Major Code: 0X0018 Minor Code: 32769 (0X8001)

Description: (OS) VioAssociate Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOASSOCIATE_POSTDT

Minor Code: 32769 (0X8001)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32770 (0X8002)

Description: (OS) VioCreateLogFont Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATELOGFONT_POSTDT

Minor Code: 32770 (0X8002)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32771 (0X8003)
OS2CHAR Major Code: 0X0018 Minor Code: 32771 (0X8003)

Description  
(OS) VioCreatePS Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATEPS_POSTDT

Minor Code  
32771 (0X8003)

Trace Groups  
VIO

Trace Types  
POST

Traced Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 32772 (0X8004)

Description  
(OS) VioDeleteSetID Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIODELETESETID_POSTDT

Minor Code  
32772 (0X8004)

Trace Groups  
VIO

Trace Types  
POST

Traced Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 32773 (0X8005)

Description  
(OS) VioDeRegister Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIODEREGISTER_POSTDT

Minor Code  
32773 (0X8005)
Trace Groups
VIO

Trace Types
POST

Traced Parameters
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32774 (0X8006)

Description
(OS) VioDestroyPS Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIODESTROYPSPSTDT

Minor Code
32774 (0X8006)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32775 (0X8007)

Description
(OS) VioEndPopUp Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOENDDPOPUP_PSTDT

Minor Code
32775 (0X8007)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w
OS2CHAR Major Code: 0X0018 Minor Code: 32776 (0X8008)

**Description**
(OS) VioGetAnsi Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETANSI_POSTDT

**Minor Code**
32776 (0X8008)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w
Indicator = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32777 (0X8009)

**Description**
(OS) VioGetBuf Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETBUF_POSTDT

**Minor Code**
32777 (0X8009)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w
Length = %w
LVBPtr = %d
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32778 (0X800A)
Description: (OS) VioGetConfig Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCONFIG_POSTDT

Minor Code: 32778 (0X800A)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

- Handle = %w
- Config Data = %r%w
- Reserved = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32779 (0X800B)

Description: (OS) VioGetCp Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCP_POSTDT

Minor Code: 32779 (0X800B)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

- Handle = %w
- Code Page Id = %w
- Reserved = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32780 (0X800C)
Description: (OS) VioGetCurPos Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURPOS_POSTDT

Minor Code: 32780 (0x800C)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

- Handle = %w
- Column = %w
- Row = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32781 (0X800D)

Description: (OS) VioGetCurType Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURTYPE_POSTDT

Minor Code: 32781 (0x800D)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

- Handle = %w
- Cursor Data = %w%w%w%w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32782 (0X800E)
**Description**
(OS) VioGetDeviceCellSize Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETDEVICECELLSIZE_POSTDT

**Minor Code**
32782 (0X800E)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Width</td>
<td>%w</td>
</tr>
<tr>
<td>Height</td>
<td>%w</td>
</tr>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 32783 (0X800F)

**Description**
(OS) VioGetFont Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETFONT_POSTDT

**Minor Code**
32783 (0X800F)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>%w</td>
</tr>
<tr>
<td>Request Block</td>
<td>%r%w</td>
</tr>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 32784 (0X8010)

**Description**
(OS) VioGetMode Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETMODE_POSTDT

**Minor Code**
32784 (0X8010)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

- Handle = %w
- ModeData = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32785 (0X8011)

**Description**
(OS) VioGetOrg Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETORG_POSTDT

**Minor Code**
32785 (0X8011)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

- PS Handle = %w
- Column = %w
- Row = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32786 (0X8012)

**Description**
(OS) VioGetPhysBuf Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPHYSBUF_POSTDT

**Minor Code**
32786 (0X8012)
Trace Groups  VIO
Trace Types   POST
Traced Parameters

DisplayBuf = %d%d%w%w
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32787 (0X8013)

Description  (OS) VioGetPSAddress Post-Invocation
Tracepoint   Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPSADDRESS_POSTDT
Minor Code   32787 (0X8013)
Trace Groups  VIO
Trace Types   POST
Traced Parameters

Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32788 (0X8014)

Description  (OS) VioGetState Post-Invocation
Tracepoint   Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETSTATE_POSTDT
Minor Code   32788 (0X8014)
Trace Groups  VIO
Trace Types   POST
Traced Parameters

Handle = %w
OS2CHAR Major Code: 0X0018 Minor Code: 32789 (0X8015)

**Description**
(OS) VioGlobalReg Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOGLOBALREG_POSTDT

**Minor Code**
32789 (0X8015)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32790 (0X8016)

**Description**
(OS) VioModeUndo Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEUNDO_POSTDT

**Minor Code**
32790 (0X8016)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32791 (0X8017)

**Description**
(OS) VioModeWait Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEWAIT_POSTDT

**Minor Code**
32791 (0X8017)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

- Notify Type = %w
- Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32792 (0X8018)

**Description**
(OS) VioPopUp Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPOPUP_POSTDT

**Minor Code**
32792 (0X8018)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

- Handle = %w
- Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32793 (0X8019)

**Description**
(OS) VioPrtSc Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSC_POSTDT

**Minor Code**
32793 (0X8019)

**Trace Groups**
VIO
Trace Types  
POST  

Traced Parameters  
Handle = %w  
Return Code = %w  

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32794 (0X801A)

Description  
(OS) VioPrtSc Toggle Post-Invocation  

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSC_TOGGLE_POSTDT  

Minor Code  
32794 (0X801A)  

Trace Groups  
VIO  

Trace Types  
POST  

Traced Parameters  
Handle = %w  
Return Code = %w  

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32795 (0X801B)

Description  
(OS) VioQueryConsole Post-Invocation  

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYCONSOLE_POSTDT  

Minor Code  
32795 (0X801B)  

Trace Groups  
VIO  

Trace Types  
POST  

Traced Parameters  
Return Code = %w
OS2CHAR Major Code: 0X0018 Minor Code: 32796 (0X801C)

**Description**
(OS) VioQueryFonts Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYFONTS_POSTDT

**Minor Code**
32796 (0X801C)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

- PS Handle = %w
- Options = %d
- Fonts Returned = %d
- Metrics Length = %d
- Metrics - szFamilyname[FACESIZE] = %s
- szFacename[FACESIZE] = %s
- idRegistry = %w
- usCodePage = %w
- lEmHeight = %d
- lXHeight = %d
- lMaxAscender = %d
- lMaxDescender = %d
- lLowerCaseAscent = %d
- lLowerCaseDescent = %d
- lInternalLeading = %d
- lExternalLeading = %d
- lAveCharWidth = %d
- lMaxCharInc = %d
- lEmInc = %d
- lMaxBaselineExt = %d
- sCharSlope = %w
- sInlineDir = %w
OS2CHAR Major Code: 0X0018 Minor Code: 32797
(0X801D)

**Description**  
(OS) VioQuerySetIDs Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYSETIDS_POSTDT

**Minor Code**  
32797 (0X801D)

**Trace Groups**  
VIO

**Trace Types**  
POST

**Traced Parameters**

- PS Handle = %w
- Count = %d
- Types = %d
- Font Names = %r%b
- Local Identifiers = %d
- Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32798 (0X801E)

**Description**  
(OS) VioReadCellStr Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCELLSTR_POSTDT

**Minor Code**  
32798 (0X801E)

**Trace Groups**  
VIO

**Trace Types**  
POST

**Traced Parameters**

- Handle = %w
- Column = %w
- Row = %w
- Length = %w
- CellStr = %r%b
OS2CHAR Major Code: 0X0018 Minor Code: 32799 (0X801F)

**Description**
(OS) VioReadCharStr Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCHARSTR_POSTDT

**Minor Code**
32799 (0X801F)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**
- Handle = %w
- Column = %w
- Row = %w
- Length = %w
- CharStr = %s%b
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32800 (0X8020)

**Description**
(OS) VioRegister Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOREGISTER_POSTDT

**Minor Code**
32800 (0X8020)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**
- Return Code = %w

________________________________________
OS2CHAR Major Code: 0X0018 Minor Code: 32801 (0X8021)

Description
(OS) VioSavRedrawUndo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWUNDO_POSTDT

Minor Code
32801 (0X8021)

Trace Groups
VIO

Trace Types
POST

Traced Parameters

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32802 (0X8022)

Description
(OS) VioSavRedrawWait Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWWAIT_POSTDT

Minor Code
32802 (0X8022)

Trace Groups
VIO

Trace Types
POST

Traced Parameters

Reserved = %w
Notify Type = %w
SavRedrawIndic = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32803 (0X8023)

Description
(OS) VioScrLock Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRLOCK_POSTDT

**Minor Code**
32803 (0X8023)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w  
Status = %b  
Wait Flag = %w  
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32804 (0X8024)

**Description**
(OS) VioScrollDn Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLDN_POSTDT

**Minor Code**
32804 (0X8024)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w  
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32805 (0X8025)

**Description**
(OS) VioScrollLf Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLLF_POSTDT

**Minor Code**
32805 (0X8025)

Trace Groups
VIO

Trace Types
POST

Traced Parameters

Handle = %w
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32806 (0X8026)

Description
(OS) VioScrollRt Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLRT_POSTDT

Minor Code
32806 (0X8026)

Trace Groups
VIO

Trace Types
POST

Traced Parameters

Handle = %w
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32807 (0X8027)

Description
(OS) VioScrollUp Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLUP_POSTDT

Minor Code
32807 (0X8027)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
OS2CHAR Major Code: 0X0018 Minor Code: 32808 (0X8028)

Description
(OS) VioScrUnLock Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRUNLOCK_POSTDT

Minor Code
32808 (0X8028)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32809 (0X8029)

Description
(OS) VioSetAnsi Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETANSI_POSTDT

Minor Code
32809 (0X8029)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32810 (0X802A)
Description  (OS) VioSetCp Post-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCP_POSTDT

Minor Code  32810 (0X802A)

Trace Groups  VIO

Trace Types  POST

Traced Parameters

Handle                   = %w
Return Code             = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32811 (0X802B)

Description  (OS) VioSetCurPos Post-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURPOS_POSTDT

Minor Code  32811 (0X802B)

Trace Groups  VIO

Trace Types  POST

Traced Parameters

Handle                   = %w
Return Code             = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32812 (0X802C)

Description  (OS) VioSetCurType Post-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURTYPE_POSTDT
Minor Code: 32812 (0X802C)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:
- Handle = %w
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32813 (0X802D)

Description: (OS) VioSetDeviceCellSize Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETDEVICECELLSIZE_POSTDT

Minor Code: 32813 (0X802D)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:
- Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32814 (0X802E)

Description: (OS) VioSetFont Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETFONT_POSTDT

Minor Code: 32814 (0X802E)

Trace Groups: VIO

Trace Types: POST
OS2CHAR Major Code: 0X0018 Minor Code: 32815 (0X802F)

Description
(OS) VioSetMode Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETMODE_POSTDT

Minor Code
32815 (0X802F)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32816 (0X8030)

Description
(OS) VioSetOrg Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETORG_POSTDT

Minor Code
32816 (0X8030)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32817 (0X8031)
Description: (OS) VioSetState Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETSTATE_POSTDT

Minor Code: 32817 (0X8031)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

Handle = %w
Return Code = %w

---------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32818 (0X8032)

Description: (OS) VioShieldInit Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDINIT_POSTDT

Minor Code: 32818 (0X8032)

Trace Groups: VIO

Trace Types: POST

Traced Parameters:

Return Code = %w

---------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32819 (0X8033)

Description: (OS) VioShieldTerm Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDTERM_POSTDT

Minor Code: 32819 (0X8033)
Trace Groups  VIO
Trace Types    POST
Traced Parameters

Return Code  = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32820 (0X8034)

Description  (OS) VioShowBuf Post-Invocation
Tracepoint   Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWBUF_POSTDT
Minor Code   32820 (0X8034)
Trace Groups VIO
Trace Types   POST
Traced Parameters

Handle  = %w
Return Code  = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32821 (0X8035)

Description  (OS) VioShowPS Post-Invocation
Tracepoint   Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWPS_POSTDT
Minor Code   32821 (0X8035)
Trace Groups VIO
Trace Types   POST
Traced Parameters

Return Code  = %w
OS2CHAR Major Code: 0X0018 Minor Code: 32822 (0X8036)

Description  
(OS) VioWrtCellStr Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCELLSTR_POSTDT

Minor Code  
32822 (0X8036)

Trace Groups  
VIO

Trace Types  
POST

Traced Parameters

Handle = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32823 (0X8037)

Description  
(OS) VioWrtCharStr Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTR_POSTDT

Minor Code  
32823 (0X8037)

Trace Groups  
VIO

Trace Types  
POST

Traced Parameters

Handle = %w
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32824 (0X8038)

Description  
(OS) VioWrtCharStrAtt Post-Invocation
**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTRCHARSTRATT_POSTDT

**Minor Code**
32824 (0X8038)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w
Return Code = %w

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 32825 (0X8039)**

**Description**
(OS) VioWrtNAttr Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTRNATTR_POSTDT

**Minor Code**
32825 (0X8039)

**Trace Groups**
VIO

**Trace Types**
POST

**Traced Parameters**

Handle = %w
Return Code = %w

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 32826 (0X803A)**

**Description**
(OS) VioWrtNCell Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTRNCCELL_POSTDT

**Minor Code**
32826 (0X803A)

**Trace Groups**
VIO
Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32827 (0X803B)

Description
(OS) VioWrtNChar Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCHAR_POSTDT

Minor Code
32827 (0X803B)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
Return Code = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 32828 (0X803C)

Description
(OS) VioWrtTTY Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTTTY_POSTDT

Minor Code
32828 (0X803C)

Trace Groups
VIO

Trace Types
POST

Traced Parameters
Handle = %w
OS2CHAR Major Code: 0X0018 Minor Code: 33025 (0X8101)

**Description**
KbdCharIn Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDCHARIN

**Minor Code**
33025 (0X8101)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**
- Major = %X
- Minor = %Y
- Return Code = %W
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W
- CharDataRec.charcode = %B
  - .scancode = %B
  - .status = %B
  - .NLSstat = %B
  - .shiftstat = %W
  - .timestamp = %D

OS2CHAR Major Code: 0X0018 Minor Code: 33026 (0X8102)

**Description**
KbdClose Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDCLOSE

**Minor Code**
33026 (0X8102)

**Trace Groups**
Trace Types
POST

Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W

---------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33027 (0X8103)

Description
KbdGetHWID Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETHWID

Minor Code
33027 (0X8103)

Trace Groups
KBD

Trace Types
POST

Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
HWIDStruc = %W
= %W

---------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33028 (0X8104)
Description: KbdDeRegister Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDDEREGISTER

Minor Code: 33028 (0X8104)

Trace Groups: KBD

Trace Types: POST

Traced Parameters:

- Major = %X
- Minor = %Y
- Return Code = %W
- ProcStatus = %B
- ProcType = %B
- SessionID = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33029 (0X8105)

Description: KbdFlushBuffer Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.KBDFLUSHBUFFER

Minor Code: 33029 (0X8105)

Trace Groups: KBD

Trace Types: POST

Traced Parameters:

- Major = %X
- Minor = %Y
- Return Code = %W
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33030 (0X8106)
**Description**
KbdFreeFocus Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDFREEFOCUS

**Minor Code**
33030 (0X8106)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**

Major = %X  Minor = %Y  
Return Code = %W 
ProcStatus = %B 
ProcType = %B  
SessionID = %W 
KbdHandle = %W

--------------------------------------------

**OS2CHAR Major Code: 0X0018 Minor Code: 33031 (0X8107)**

**Description**
KbdGetCP Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETCP

**Minor Code**
33031 (0X8107)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**

Major = %X  Minor = %Y  
Return Code = %W 
ProcStatus = %B 
ProcType = %B  
SessionID = %W 
KbdHandle = %W
OS2CHAR Major Code: 0X0018 Minor Code: 33032 (0X8108)

**Description**
KbdGetFocus Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETFOCUS

**Minor Code**
33032 (0X8108)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**
- Major = %X  Minor = %Y
- Return Code = %W
- ProcStatus = %B
- ProcType = %B
- SessionID = %W
- KbdHandle = %W

OS2CHAR Major Code: 0X0018 Minor Code: 33033 (0X8109)

**Description**
KbdGetStatus Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETSTATUS

**Minor Code**
33033 (0X8109)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**
- Major = %X  Minor = %Y
- Return Code = %W
OS2CHAR Major Code: 0X0018 Minor Code: 33034 (0X810A)

**Description**  
KbdOpen Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDOPEN

**Minor Code**  
33034 (0X810A)

**Trace Groups**  
KBD

**Trace Types**  
POST

**Traced Parameters**

- Major = %X  
- Minor = %Y  
- Return Code = %W  
- ProcStatus = %B  
- ProcType = %B  
- SessionID = %W  
- KbdHandle = %W

OS2CHAR Major Code: 0X0018 Minor Code: 33035 (0X810B)

**Description**  
KbdPeek Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDPEEK
OS2CHAR Major Code: 0X0018 Minor Code: 33036 (0X810C)

Description
KbdRegister Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDREGISTER

Minor Code
33036 (0X810C)

Trace Groups
KBD

Trace Types
POST

Traced Parameters

Major = %X Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
CharDataRec.charcode = %B
.scancode = %B
.status = %B
.NLSstat = %B
.shiftstat = %W
.timestamp = %D
OS2CHAR Major Code: 0X0018 Minor Code: 33037 (0X810D)

Description  
KbdSetCP Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCUSTXT

Minor Code  
33037 (0X810D)

Trace Groups  
KBD

Trace Types  
POST

Traced Parameters  
Major = %X  Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W

OS2CHAR Major Code: 0X0018 Minor Code: 33038 (0X810E)

Description  
KbdSetCustXT Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCUSTXT

Minor Code  
33038 (0X810E)

Trace Groups  
KBD

Trace Types  
POST

Traced Parameters  
Major = %X  Minor = %Y
Return Code = %W  
ProcStatus = %B  
ProcType   = %B  
SessionID = %W  
KbdHandle = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33039 (0X810F)

**Description**  
KbdSetFgnd Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETFGND

**Minor Code**  
33039 (0X810F)

**Trace Groups**  
KBD

**Trace Types**  
POST

**Traced Parameters**

Major = %X   Minor = %Y  
Return Code = %W  
ProcStatus = %B  
ProcType   = %B  
SessionID = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33040 (0X8110)

**Description**  
KbdSetStatus Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETSTATUS

**Minor Code**  
33040 (0X8110)

**Trace Groups**  
KBD

**Trace Types**  
POST
Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33041 (0X8111)

Description
KbdShellInit Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBSDHELLINIT

Minor Code
33041 (0X8111)

Trace Groups
KBD

Trace Types
POST

Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33042 (0X8112)

Description
KbdStringIn Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSTRINGIN

Minor Code
33042 (0X8112)

Trace Groups
KBD
Trace Types

POST

Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W
KbdHandle = %W
CharBuffer length = %W
CharBuffer = %S

Note: CharBuffer was added with OS/2 Warp V3.0 fix pack 41 and OS/2 Warp V4.0 fix pack 10.

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33043 (0X8113)

Description
KbdSynch Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.KBDSYNCH

Minor Code
33043 (0X8113)

Trace Groups
KBD

Trace Types
POST

Traced Parameters

Major = %X   Minor = %Y
Return Code = %W
ProcStatus = %B
ProcType = %B
SessionID = %W

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33044 (0X8114)
**Description**
KbdXlate Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.KBDXLATE

**Minor Code**
33044 (0X8114)

**Trace Groups**
KBD

**Trace Types**
POST

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major = %X</td>
<td></td>
<td>Minor = %Y</td>
</tr>
<tr>
<td>Return Code = %W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcStatus = %B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcType = %B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SessionID = %W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KbdHandle = %W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XlateRecord = %B</td>
<td></td>
<td>%B %B %B %B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%B %B %B %B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%B %B %B %B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%B %B %B</td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0X0018 Minor Code: 33281 (0X8201)

**Description**
(MOU) MouClose Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUCLOSE

**Minor Code**
33281 (0X8201)

**Trace Groups**
MOU

**Trace Types**
POST

**Traced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Code = %w</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcStatus = %b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcType = %b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OS2CHAR Major Code: 0X0018 Minor Code: 33282 (0X8202)

**Description**
(MOU) MouDeRegister Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUDEREGISTER

**Minor Code**
33282 (0X8202)

**Trace Groups**
MOU

**Trace Types**
POST

**Traced Parameters**
Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w

OS2CHAR Major Code: 0X0018 Minor Code: 33283 (0X8203)

**Description**
(MOU) MouDrawPtr Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: OS2CHAR.MOUDRAWPTR

**Minor Code**
33283 (0X8203)

**Trace Groups**
MOU

**Trace Types**
POST

**Traced Parameters**
Return Code = %w
ProcStatus = %b
ProcType = %b
OS2CHAR Major Code: 0X0018 Minor Code: 33284 (0X8204)

Description
(MOU) MouFlushQue Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUFLUSHQUE

Minor Code
33284 (0X8204)

Trace Groups
MOU

Trace Types
POST

Traced Parameters
Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

OS2CHAR Major Code: 0X0018 Minor Code: 33285 (0X8205)

Description
(MOU) MouGetDevStatus Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETDEVSTATUS

Minor Code
33285 (0X8205)

Trace Groups
MOU

Trace Types
POST

Traced Parameters
Return Code = %w
ProcStatus = %b
OS2CHAR Major Code: 0X0018 Minor Code: 33286 (0X8206)

Description: (MOU) MouGetEventMask Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETEVENTMASK

Minor Code: 33286 (0X8206)

Trace Groups: MOU

Trace Types: POST

Traced Parameters:

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
EventMask = %w

OS2CHAR Major Code: 0X0018 Minor Code: 33287 (0X8207)

Description: (MOU) MouGetNumButtons Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMBUTTONS

Minor Code: 33287 (0X8207)

Trace Groups: MOU

Trace Types: POST
Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
NumButtons = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33288 (0X8208)

Description
(MOU) MouGetNumMickey's Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMMICKEYS

Minor Code
33288 (0X8208)

Trace Groups
MOU

Trace Types
POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
NumMickey = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33289 (0X8209)

Description
(MOU) MouGetNumQueEI Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMQUEEL

Minor Code
33289 (0X8209)
Trace Groups  MOU
Trace Types  POST
Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType   = %b
SessionID = %w
MouHandle = %w
Que Events = %w
Que Max Events = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33290 (0X820A)

Description  (MOU) MouGetPtrPos Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRPOS
Minor Code  33290 (0X820A)
Trace Groups  MOU
Trace Types  POST
Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType   = %b
SessionID = %w
MouHandle = %w
PtrPos.ROW = %w
.COL = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33291 (0X820B)
Description  (MOU) MouGetPtrShape Post-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRSHAPE

Minor Code  33291 (0X820B)

Trace Groups  MOU

Trace Types  POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
PtrDefRec.LENGTH = %w
.COL = %w
.ROW = %w
.COLOFFSET = %w
.ROWOFFSET = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33292 (0X820C)

Description  (MOU) MouGetScaleFact Post-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETSCALEFACT

Minor Code  33292 (0X820C)

Trace Groups  MOU

Trace Types  POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
OS2CHAR Major Code: 0X0018 Minor Code: 33293 (0X820D)

Description
(MOU) MouGetThreshold Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETTHRESHOLD

Minor Code
33293 (0X820D)

Trace Groups
MOU

Trace Types
POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w
ThresholdStruc.Length = %w
.Level1 = %w
.Lev1Mult = %w
.Level2 = %w
.Lev2Mult = %w

OS2CHAR Major Code: 0X0018 Minor Code: 33294 (0X820E)

Description
(MOU) MouInitReal Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: OS2CHAR.MOUINITREAL
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>33294 (0x820E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MOU</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>Return Code = %w</td>
<td></td>
</tr>
<tr>
<td>ProcStatus = %b</td>
<td></td>
</tr>
<tr>
<td>ProcType = %b</td>
<td></td>
</tr>
<tr>
<td>SessionID = %w</td>
<td></td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0x0018 Minor Code: 33295 (0x820F)

Description | (MOU) MouOpen Post-Invocation
Tracepoint  | Public symbol defined dynamic tracepoint: OS2CHAR.MOUOPEN

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>33295 (0x820F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MOU</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>Return Code = %w</td>
<td></td>
</tr>
<tr>
<td>ProcStatus = %b</td>
<td></td>
</tr>
<tr>
<td>ProcType = %b</td>
<td></td>
</tr>
<tr>
<td>SessionID = %w</td>
<td></td>
</tr>
<tr>
<td>MouHandle = %w</td>
<td></td>
</tr>
</tbody>
</table>

OS2CHAR Major Code: 0x0018 Minor Code: 33296 (0x8210)

Description | (MOU) MouReadEventQue Post-Invocation
Tracepoint  | Public symbol defined dynamic tracepoint: OS2CHAR.MOUREADEVENTQUE
Minor Code: 33296 (0x8210)
Trace Groups: MOU
Trace Types: POST
Traced Parameters:

- Return Code = %w
- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w
- EventBuffer.MOUSTATE = %w
- .EVENTTIME = %d
- .ROW = %w
- .COL = %w
- ReadType = %w

-------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33297 (0X8211)

Description:
(MOU) MouRegister Post-Invocation

Tracepoint:
Public symbol defined dynamic tracepoint: OS2CHAR.MOUREGISTER

Minor Code: 33297 (0X8211)
Trace Groups: MOU
Trace Types: POST
Traced Parameters:

- Return Code = %w
- ProcStatus = %b
- ProcType = %b
- SessionID = %w

-------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33298 (0X8212)
Description: (MOU) MouRemovePtr Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUREMOVEPTR

Minor Code: 33298 (0X8212)

Trace Groups: MOU

Trace Types: POST

Traced Parameters:

- Return Code = %w
- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33299 (0X8213)

Description: (MOU) MouSetDevStatus Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETDEVSTATUS

Minor Code: 33299 (0X8213)

Trace Groups: MOU

Trace Types: POST

Traced Parameters:

- Return Code = %w
- ProcStatus = %b
- ProcType = %b
- SessionID = %w
- MouHandle = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33300 (0X8214)
MOU Major Code: 0X0018 Minor Code: 33300 (0X8214)

Description  (MOU) MouSetEventMask Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETEVENTMASK
Minor Code  33300 (0X8214)
Trace Groups  MOU
Trace Types  POST
Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

-----------------------------

MOU Major Code: 0X0018 Minor Code: 33301 (0X8215)

Description  (MOU) MouSetPtrPos Post-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETPTRPOS
Minor Code  33301 (0X8215)
Trace Groups  MOU
Trace Types  POST
Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

-----------------------------

MOU Major Code: 0X0018 Minor Code: 33302 (0X8216)
(MOU) MouSetPtrShape Post-Invocation

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETPTRSHAPE

Minor Code 33302 (0X8216)

Trace Groups MOU

Trace Types POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33303 (0X8217)

(MOU) MouSetScaleFact Post-Invocation

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETSCALEFACT

Minor Code 33303 (0X8217)

Trace Groups MOU

Trace Types POST

Traced Parameters

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33304 (0X8218)
Description: (MOU) MouSetThreshold Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUSETTHRESHOLD

Minor Code: 33304 (0X8218)

Trace Groups: MOU

Trace Types: POST

Traced Parameters:

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w
MouHandle = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33305 (0X8219)

Description: (MOU) MouShellInit Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: OS2CHAR.MOUSSHILLINIT

Minor Code: 33305 (0X8219)

Trace Groups: MOU

Trace Types: POST

Traced Parameters:

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w

--------------------------------------------

OS2CHAR Major Code: 0X0018 Minor Code: 33306 (0X821A)
(MOU) MouSynch Post-Invocation

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSYNCH

33306 (0X821A)

MOU

POST

Return Code = %w
ProcStatus = %b
ProcType = %b
SessionID = %w

--------------------------------------------

QUECALLS.DLL Trace Events

The tracepoints for the QUECALLS.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace Events for QUECALLS Major Code: 0X0016, Sorted by Minor Code

QUECALLS API Tracepoints Indirected Via DOSCALL1.

--------------------------------------------

Trace Events for QUECALLS Major Code: 0X0016, Sorted by Minor Code

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>(OS) DosCloseQueue Pre-Invocation</td>
</tr>
<tr>
<td>00002</td>
<td>(OS) DosCreateQueue Pre-Invocation</td>
</tr>
<tr>
<td>00003</td>
<td>(OS) DosOpenQueue Pre-Invocation</td>
</tr>
<tr>
<td>00004</td>
<td>(OS) DosPeerQueue Pre-Invocation</td>
</tr>
<tr>
<td>00005</td>
<td>(OS) DosPurgeQueue Pre-Invocation</td>
</tr>
<tr>
<td>00006</td>
<td>(OS) DosQueryQueue Pre-Invocation</td>
</tr>
<tr>
<td>00007</td>
<td>(OS) DosReadQueue Pre-Invocation</td>
</tr>
<tr>
<td>00008</td>
<td>(OS) DosWriteQueue Pre-Invocation</td>
</tr>
<tr>
<td>00009</td>
<td>(OS) Peek Data Packet From Queue</td>
</tr>
<tr>
<td>00010</td>
<td>(OS) Read Data Packet From Queue</td>
</tr>
<tr>
<td>00011</td>
<td>(OS) Write Data Packet To Queue</td>
</tr>
<tr>
<td>32769</td>
<td>(OS) DosCloseQueue Post-Invocation</td>
</tr>
<tr>
<td>32770</td>
<td>(OS) DosCreateQueue Post-Invocation</td>
</tr>
</tbody>
</table>
Trace Events for QUECALLS Major Code: 0X0016, Sorted by Tracepoint

(OS) DosCloseQueue Post-Invocation 32769 (0X8001)
(OS) DosCloseQueue Pre-Invocation 00001 (0X0001)
(OS) DosCreateQueue Post-Invocation 32770 (0X8002)
(OS) DosCreateQueue Pre-Invocation 00002 (0X0002)
(OS) DosOpenQueue Post-Invocation 32771 (0X8003)
(OS) DosOpenQueue Pre-Invocation 00003 (0X0003)
(OS) DosPeekQueue Post-Invocation 32772 (0X8004)
(OS) DosPeekQueue Pre-Invocation 00004 (0X0004)
(OS) DosPurgeQueue Post-Invocation 32773 (0X8005)
(OS) DosPurgeQueue Pre-Invocation 00005 (0X0005)
(OS) DosQueryQueue Post-Invocation 32774 (0X8006)
(OS) DosQueryQueue Pre-Invocation 00006 (0X0006)
(OS) DosReadQueue Post-Invocation 32775 (0X8007)
(OS) DosReadQueue Pre-Invocation 00007 (0X0007)
(OS) DosWriteQueue Post-Invocation 32776 (0X8008)
(OS) DosWriteQueue Pre-Invocation 00008 (0X0008)
(OS) Peek Data Packet From Queue 00009 (0X0009)
(OS) Read Data Packet From Queue 00010 (0X000A)
(OS) Write Data Packet To Queue 00011 (0X000B)

QUECALLS Major Code: 0X0016 Minor Code: 1 (0X0001)

Description
(OS) DosCloseQueue Pre-Invocation

Tracepoint
Source line defined dynamic tracepoint: @closeq.c in QUECALLS.

Minor Code
1 (0X0001)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters
Handle=%w

QUECALLS Major Code: 0X0016 Minor Code: 2 (0X0002)
**Description**
(OS) DosCreateQueue Pre-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @createq.c in QUECALLS.

**Minor Code**
2 (0X0002)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Priority=%w Queue name=%s

--------------------------------------------

**QUECALLS Major Code: 0X0016 Minor Code: 3 (0X0003)**

**Description**
(OS) DosOpenQueue Pre-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @openq.c in QUECALLS.

**Minor Code**
3 (0X0003)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Queue name=%s

--------------------------------------------

**QUECALLS Major Code: 0X0016 Minor Code: 4 (0X0004)**

**Description**
(OS) DosPeekQueue Pre-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @peekq.c in QUECALLS.

**Minor Code**
4 (0X0004)

**Trace Groups**
No groups assigned.
Trace Types
PRE

Traced Parameters
Handle=%w Element Code=%w
Semaphore Handle=%d No Wait Flag=%b%i1

QUECALLS Major Code: 0X0016 Minor Code: 5 (0X0005)

Description
(OS) DosPurgeQueue Pre-Invocation

Tracepoint
Source line defined dynamic tracepoint: @purgeq.c in QUECALLS.

Minor Code
5 (0X0005)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters
Handle=%w

QUECALLS Major Code: 0X0016 Minor Code: 6 (0X0006)

Description
(OS) DosQueryQueue Pre-Invocation

Tracepoint
Source line defined dynamic tracepoint: @queryq.c in QUECALLS.

Minor Code
6 (0X0006)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters
Handle=%w
QUECALLS Major Code: 0X0016 Minor Code: 7 (0X0007)

**Description**
(OS) DosReadQueue Pre-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @readq.c in QUECALLS.

**Minor Code**
7 (0X0007)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Handle=%w Element Code=%w
Semaphore Handle=%d No Wait Flag=%b

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 8 (0X0008)

**Description**
(OS) DosWriteQueue Pre-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @writeq.c in QUECALLS.

**Minor Code**
8 (0X0008)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Handle=%w Request=%d Length=%d
Address=%a Priority=%b

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 9 (0X0009)

**Description**
(OS) Peek Data Packet From Queue

**Tracepoint**
Source line defined dynamic tracepoint: @peekq.c in QUECALLS.

**Minor Code**
9 (0X0009)

**Trace Groups**
No groups assigned.

**Trace Types**
INT

**Traced Parameters**
Length=%w Address=%a Element Code=%w

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 10 (0X000A)

**Description**
(OS) Read Data Packet From Queue

**Tracepoint**
Source line defined dynamic tracepoint: @readq.c in QUECALLS.

**Minor Code**
10 (0X000A)

**Trace Groups**
No groups assigned.

**Trace Types**
INT

**Traced Parameters**
Length=%d Address=%a Element Code=%w

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 11 (0X000B)

**Description**
(OS) Write Data Packet To Queue

**Tracepoint**
Source line defined dynamic tracepoint: @writeq.c in QUECALLS.

**Minor Code**
11 (0X000B)

**Trace Groups**
No groups assigned.

**Trace Types**
INT

**Traced Parameters**
QUECALLS Major Code: 0X0016 Minor Code: 32769 (0X8001)

**Description**
(OS) DosCloseQueue Post-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @closeq.c in QUECALLS.

**Minor Code**
32769 (0X8001)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Return Code=%w

QUECALLS Major Code: 0X0016 Minor Code: 32770 (0X8002)

**Description**
(OS) DosCreateQueue Post-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @createq.c in QUECALLS.

**Minor Code**
32770 (0X8002)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Return Code=%w Handle=%w

QUECALLS Major Code: 0X0016 Minor Code: 32771

(0X8003)

**Description**
(OS) DosOpenQueue Post-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @openq.c in QUECALLS.

**Minor Code**
32771 (0X8003)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Return Code=%w Handle=%w Owner PID=%w

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 32772 (0X8004)

**Description**
(OS) DosPeekQueue Post-Invocation

**Tracepoint**
Source line defined dynamic tracepoint: @peekq.c in QUECALLS.

**Minor Code**
32772 (0X8004)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**
Return Code=%w Request=%w%d Address=%a Length=%d
Element Code=%w Priority=%b

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 32773 (0X8005)

**Description**
(OS) DosPurgeQueue Post-Invocation
Tracepoint
Source line defined dynamic tracepoint: @purgeq.c in QUECALLS.

Minor Code
32773 (0X8005)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code=%w

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 32774 (0X8006)

Description
(OS) DosQueryQueue Post-Invocation

Tracepoint
Source line defined dynamic tracepoint: @queryq.c in QUECALLS.

Minor Code
32774 (0X8006)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code=%w Number of Elements=%w

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 32775 (0X8007)

Description
(OS) DosReadQueue Post-Invocation

Tracepoint
Source line defined dynamic tracepoint: @readq.c in QUECALLS.

Minor Code
32775 (0X8007)

Trace Groups
No groups assigned.
Trace Types
POST

Traced Parameters
Return Code=%w Request=%w%d Address=%a Length=%d
Priority=%b

--------------------------------------------

QUECALLS Major Code: 0X0016 Minor Code: 32776 (0X8008)

Description
(OS) DosWriteQueue Post-Invocation

Tracepoint
Source line defined dynamic tracepoint: @writeq.c in QUECALLS.

Minor Code
32776 (0X8008)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters
Return Code=%w

--------------------------------------------

SESMGR.DLL Trace Events

The tracepoints for the SESMGR.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for SESMGR Major Code: 0X0017, sorted by minor code.
Trace events for SESMGR Major Code: 0X0017, sorted by tracepoint.
Indirected Session Manager API Tracepoints.

--------------------------------------------

Trace Events for SESMGR Major Code: 0X0017, Sorted by Minor Code
00001 (0X0001) (OS) DosSelectSession Pre-Invocation
00002 (0X0002) (OS) DosSetSession Pre-Invocation
00004 (0X0004) (OS) DosSMAppNotify Pre-Invocation
00005 (0X0005) (OS) DosSMChidExit Pre-Invocation
00007 (0X0007) (OS) DosSMDoAppReq Pre-Invocation
00008 (0X0008) (OS) DosSMFreeSglId Pre-Invocation
00009 (0X0009) (OS) DosSMGetSglId Pre-Invocation
00011 (0X000B) (OS) DosSMNotifyDD Pre-Invocation
00012 (0X000C) (OS) DosSMNotifyDD2 Pre-Invocation
00013 (0X000D) (OS) DosSMParentSwitch Pre-Invocation
00014 (0X000E) (OS) DosSMGSoPopup Pre-Invocation
00015 (0X000F) (OS) DosSMGEndPopup Pre-Invocation
00016 (0X0010) (OS) DosSMGSetPre-Invocation
00017 (0X0011) (OS) DosSMGStart Pre-Invocation
00018 (0X0012) (OS) DosSMGSwitch Pre-Invocation
00019 (0X0013) (OS) DosSMGTruncate Pre-Invocation
00020 (0X0014) (OS) DosStartSession Pre-Invocation
00021 (0X0015) (OS) DosStopSession Pre-Invocation
00023 (0X0017) (OS) ParentNotify Pre-Invocation
00024 (0X0018) (OS) ParentSwitch Pre-Invocation
00025 (0X0019) (OS) WriteTermQueue Pre-Invocation
32769 (0X8001) (OS) DosSelectSession Post-Invocation
32770 (0X8002) (OS) DosSetSession Post-Invocation
32772 (0X8004) (OS) DosSMAppNotify Post-Invocation
32773 (0X8005) (OS) DosSMChidExit Post-Invocation
32775 (0X8007) (OS) DosSMDoAppReq Post-Invocation
32779 (0X800B) (OS) DosSMNotifyDD Post-Invocation
32780 (0X800C) (OS) DosSMNotifyDD2 Post-Invocation
32781 (0X800D) (OS) DosSMGStart Post-Invocation
32782 (0X800E) (OS) DosSMGEndPopup Post-Invocation
32783 (0X800F) (OS) DosSMGSetPost-Invocation
32784 (0X8010) (OS) DosSMGSwitch Post-Invocation
32785 (0X8011) (OS) DosSMGTruncate Post-Invocation
32786 (0X8012) (OS) DosSMGStart Post-Invocation
32787 (0X8013) (OS) DosSMGTruncate Post-Invocation
32788 (0X8014) (OS) DosStartSession Post-Invocation
32789 (0X8015) (OS) DosStopSession Post-Invocation
32791 (0X8017) (OS) ParentNotify Post-Invocation
32792 (0X8018) (OS) ParentSwitch Post-Invocation
32793 (0X8019) (OS) WriteTermQueue Post-Invocation

Trace Events for SESMGR Major Code: 0X0017, Sorted by Tracepoint

DOSSELECTSESSION 00001 (0X0001)
DOSSELECTSESSION_POSTDT 32769 (0X8001)
DOSMADDSEQQUEUE 00002 (0X0002)
DOSMADDSEQQUEUE_POSTDT 32770 (0X8002)
DOSMAPPNOTIFY 00004 (0X0004)
DOSMAPPNOTIFY_POSTDT 32772 (0X8004)
DOSMDELSSEQQUEUE 00005 (0X0005)
DOSMDELSSEQQUEUE_POSTDT 32773 (0X8005)
DOSMDOPREQ 00007 (0X0007)
DOSMFREESESSID 00008 (0X0008)
DOSMGTSSEQQUEUE 00009 (0X0009)
DOSMGTSSEQQUEUE_POSTDT 32775 (0X8007)
DOSMGGETSTATUS 00021 (0X0015)
DOSMGGETSTATUS_POSTDT 32789 (0X8015)
DOSMNOTIFYDD 00011 (0X000B)
DOSMNOTIFYDD2 00012 (0X000C)
DOSMNOTIFYDD_POSTDT 32780 (0X800C)
DOSMNOTIFYDD_POSTDT 32779 (0X800B)
DOSMPARENTSWITCH 00013 (0X000D)
DOSMPARENTSWITCH_POSTDT 32781 (0X800D)
SESMGR Major Code: 0X0017 Minor Code: 1 (0X0001)

**Description**
(OS) DosSelectSession Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSELECTSESSION

**Minor Code**
1 (0X0001)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
Reserved = %d
Select Option = %w

SESMGR Major Code: 0X0017 Minor Code: 2 (0X0002)

**Description**
(OS) DosSetSession Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMADDSGQUEUE

**Minor Code**
2 (0X0002)

**Trace Groups**
No groups assigned.
Trace Types
PRE

Traced Parameters
Status Data - SetLength = %w
SelectOpt = %w
BondOption = %w
Session Id = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 4 (0X0004)

Description
(OS) DosSMAppNotify Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMAPPNOTIFY

Minor Code
4 (0X0004)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters
Sessions Return Code = %w
Terminated Session Id = %w
Notify Switch Action = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 5 (0X0005)

Description
(OS) DosSMChildExit Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMDELSGQUEUE

Minor Code
5 (0X0005)

Trace Groups
No groups assigned.

Trace Types
PRE
Traced Parameters

SGID = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 7 (0X0007)

Description (OS) DosSMDoAppReq Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: SESMGR.DOSSMDOAPPLE
Minor Code 7 (0X0007)
Trace Groups No groups assigned.
Trace Types PRE
Traced Parameters

Request Header = %r%b
Request Data = %r%b

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 8 (0X0008)

Description (OS) DosSMFreeSGId Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: SESMGR.DOSSMDFREESGID
Minor Code 8 (0X0008)
Trace Groups No groups assigned.
Trace Types PRE
Traced Parameters

SGID = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 9 (0X0009)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMGetSGId Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSGQUEUE</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>9 (0X0009)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMNotifyDD Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>11 (0X000B)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>Outgoing SG = %w</td>
</tr>
<tr>
<td></td>
<td>Incoming SG = %w</td>
</tr>
<tr>
<td></td>
<td>Notification Type = %w</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMNotifyDD2 Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD2</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>12 (0X000C)</td>
</tr>
</tbody>
</table>
Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters: None

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 13 (0X000D)

Description: (OS) DosSMParentSwitch Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH

Minor Code: 13 (0X000D)

Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters:

Session Id = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 14 (0X000E)

Description: (OS) DosSMSGDoPopup Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGDOPOPUP

Minor Code: 14 (0X000E)

Trace Groups: No groups assigned.

Trace Types: PRE

Traced Parameters:

SGID = %w

PID = %w
SESMGR Major Code: 0X0017 Minor Code: 15 (0X000F)

**Description**
(OS) DosSMSGEndPopup Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGENDPOPUP

**Minor Code**
15 (0X000F)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**
None

SESMGR Major Code: 0X0017 Minor Code: 16 (0X0010)

**Description**
(OS) DosSMSGSet Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSET

**Minor Code**
16 (0X0010)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**

Bond Option = %w
Select Option = %w
Child Session Id = %w
Parent Session Id = %w

SESMGR Major Code: 0X0017 Minor Code: 17 (0X0011)
Description (OS) DosSMSSGStart Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSTART

Minor Code 17 (0X0011)

Trace Groups No groups assigned.

Trace Types PRE

Traced Parameters

Reserved = %d
Debug PID = %w
Debug SID = %w
PM Struct = %d
Request Block = %d
Environ Option = %w
New Process Id = %w
New Session = %w
Parent Id = %w
Program Inputs = %s %b
Program Name = %s %b
Program Title = %s
Program Type = %w
Asynchronous Option = %w
Start Mode = %w
Save Action = %w

-------------------------------

SESMGR Major Code: 0X0017 Minor Code: 18 (0X0012)

Description (OS) DosSMSSGSwitch Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSWITCH

Minor Code 18 (0X0012)

Trace Groups
Trace Types
PRE

Traced Parameters

Reserved = %d
Debug SIG = %w
Session Id = %w
Switch Action = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 19 (0X0013)

Description
(OS) DosSMSGTerminate Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGTERMINATE

Minor Code
19 (0X0013)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters

Reserved = %d
Session Id = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 20 (0X0014)

Description
(OS) DosStartSession Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSTARTSESSION

Minor Code
20 (0X0014)

Trace Groups
No groups assigned.

Trace Types
PRE

Traced Parameters
SESMGR Major Code: 0X0017 Minor Code: 21 (0X0015)

**Description**
(OS) DosStopSession Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSTATUS

**Minor Code**
21 (0X0015)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**

Reserved = %d
Session Id = %w
Stop Session Option = %w

SESMGR Major Code: 0X0017 Minor Code: 23 (0X0017)

**Description**
(OS) ParentNotify Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.PARENTNOTIFY

**Minor Code**
23 (0X0017)

**Trace Groups**
No groups assigned.

**Trace Types**
PRE

**Traced Parameters**

Session Id = %w
TermType = %w

SESMGR Major Code: 0X0017 Minor Code: 24 (0X0018)
<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) ParentSwitch Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: SESMGR.PARENTSWITCH</td>
</tr>
<tr>
<td>Minor Code</td>
<td>24 (0X0018)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Session Id = %w</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) WriteTermQueue Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: SESMGR.WRITETERMQUEUE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>25 (0X0019)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Action = %w, Parent Process Id = %w, Parent Screen Group = %w, Terminating Session SID = %w, PID = %w</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSelectSession Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32769 (0X8001)</td>
</tr>
</tbody>
</table>

| Description                  | (OS) DosSelectSession Post-Invocation                                  |
Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSELECTSESSION_POSTDT

Minor Code
32769 (0X8001)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters
Return Code = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32770 (0X8002)

Description
(OS) DosSetSession Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMADDSQUEUE_POSTDT

Minor Code
32770 (0X8002)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32772 (0X8004)

Description
(OS) DosSMAppNotify Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMAPPNOTIFY_POSTDT

Minor Code
32772 (0X8004)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

--------------------------------------------
Return Code = %w

SESMGR Major Code: 0X0017 Minor Code: 32773 (0X8005)

Description
(OS) DosSMChildExit Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMDELSGQUEUE_POSTDT

Minor Code
32773 (0X8005)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code = %w

SESMGR Major Code: 0X0017 Minor Code: 32775 (0X8007)

Description
(OS) DosSMDoAppReq Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSGQUEUE_POSTDT

Minor Code
32775 (0X8007)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code = %w

SESMGR Major Code: 0X0017 Minor Code: 32779 (0X800B)

Description
(OS) DosSMNotifyDD Post-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32779 (0X800B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

Return Code = %w

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMNotifyDD2 Post-Invocation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD2_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32780 (0X800C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

DosDevIOCTLsRC = %w

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMParentSwitch Post-Invocation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32781 (0X800D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMParentSwitch Post-Invocation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32781 (0X800D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMParentSwitch Post-Invocation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32781 (0X800D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>(OS) DosSMParentSwitch Post-Invocation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH_POSTDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>32781 (0X800D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
**SESMGR Major Code: 0X0017 Minor Code: 32782 (0X800E)**

**Description**
(OS) DosSMSGDoPopup Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGDOPOPUP_POSTDT

**Minor Code**
32782 (0X800E)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

**SESMGR Major Code: 0X0017 Minor Code: 32783 (0X800F)**

**Description**
(OS) DosSMSGEndPopup Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGENDPOPUP_POSTDT

**Minor Code**
32783 (0X800F)

**Trace Groups**
No groups assigned.

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

**SESMGR Major Code: 0X0017 Minor Code: 32784 (0X8010)**

**Description**
**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSET_POSTDT

**Minor Code**

32784 (0X8010)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code \( = \%w \)

--------------------------------------------

**Description**

(OS) DosSMSGStart Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSTART_POSTDT

**Minor Code**

32785 (0X8011)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code \( = \%w \)

--------------------------------------------

**Description**

(OS) DosSMSGSwitch Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSWITCH_POSTDT

**Minor Code**

32786 (0X8012)

**Trace Groups**

No groups assigned.

**Trace Types**

POST
Traced Parameters

Return Code = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32787 (0X8013)

Description
(OS) DosSMSGTerminate Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGTERMINATE_POSTDT

Minor Code
32787 (0X8013)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32788 (0X8014)

Description
(OS) DosStartSession Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: SESMGR.DOSSTARTSESSION_POSTDT

Minor Code
32788 (0X8014)

Trace Groups
No groups assigned.

Trace Types
POST

Traced Parameters

New Process Id = %w
New Session = %w
Return Code = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32789 (0X8015)
**Description**  (OS) DosStopSession Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSTATUS_POSTDT

**Minor Code**  32789 (0X8015)

**Trace Groups**  No groups assigned.

**Trace Types**  POST

**Traced Parameters**

Return Code  = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32791 (0X8017)

**Description**  (OS) ParentNotify Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: SESMGR.PARENTNOTIFY_POSTDT

**Minor Code**  32791 (0X8017)

**Trace Groups**  No groups assigned.

**Trace Types**  POST

**Traced Parameters**

Switched  = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32792 (0X8018)

**Description**  (OS) ParentSwitch Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: SESMGR.PARENTSWITCH_POSTDT

**Minor Code**  32792 (0X8018)

**Trace Groups**  No groups assigned.
Trace Types

POST

Traced Parameters

Switched = %w

--------------------------------------------

SESMGR Major Code: 0X0017 Minor Code: 32793 (0X8019)

Description

(OS) WriteTermQueue Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.WRITETERMQUEUE_POSTDT

Minor Code

32793 (0X8019)

Trace Groups

No groups assigned.

Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

Indirected Session Manager API Tracepoints

The following table lists 32-bit pre-invocation tracepoints for SESMGR.DLL APIs that are indirected via SESMGR thunking layer. These should be trace in conjunction with their corresponding SESMGR 16-bit API tracepoints.

<table>
<thead>
<tr>
<th>SESMGR API</th>
<th>Minor code</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS32SELECTSESSION</td>
<td>32</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32SETSESSION</td>
<td>33</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32STARTSESSION</td>
<td>34</td>
<td>PRE, API</td>
</tr>
<tr>
<td>DOS32STOPSESSION</td>
<td>35</td>
<td>PRE, API</td>
</tr>
</tbody>
</table>

--------------------------------------------

Multi-Media Extensions

The tracepoints for the Multi-Media Extensions major code are identified in the following tables. These tracepoints are static tracepoints.
Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for Multi-Media Extensions major code: 0X006D, sorted by minor code.
Trace events for Multi-Media Extensions major code: 0X006D, sorted by tracepoint.

--------------------------------------------

Trace Events for Multi-Media Extensions Major Code: 0X006D, Sorted by Minor Code

00001 (0X0001) SSM_ShcAssociate_Entry
00002 (0X0002) SSM_ShcClose_Entry
00003 (0X0003) SSM_ShcCreate_Entry
00004 (0X0004) SSM_ShcDestroy_Entry
00005 (0X0005) SSM_ShcStart_Entry
00006 (0X0006) SSM_ShcStop_Entry
00007 (0X0007) SSM_ShcSeek_Entry
00008 (0X0008) SSM_ShcEnableEvt_Entry
00009 (0X0009) SSM_ShcDisableEvt_Entry
00010 (0X000A) SSM_ShcEnableSync_Entry
00011 (0X000B) SSM_ShcDisableSync_Entry
00012 (0X000C) SSM_ShcGetTime_Entry
00013 (0X000D) SSM_ShcGetProtocol_Entry
00014 (0X000E) SSM_ShcInstProtocol_Entry
00015 (0X000F) SSM_ShcEnumProtocol_Entry
00016 (0X0010) SSM_ShcNegotResult_Entry
00017 (0X0011) SSM_ShcSendMsg_Entry
00018 (0X0012) UNUSED_HOOK_109_018
00019 (0X0013) UNUSED_HOOK_109_019
00020 (0X0014) SSM_ProcRun_Entry
00021 (0X0015) SSM_ProcBlock_Entry
00022 (0X0016) SSM_ioctl_Entry
00023 (0X0017) SSM_IDC_Call_To_SHC
00024 (0X0018) SSM_IDCNotifyEntry
00025 (0X0019) SSM_IDCDeRegEntry
00026 (0X001A) SSM_IDCRptEvtEntry
00027 (0X001B) GET_EMPTY_Entry
00028 (0X001C) GET_FULL_Entry
00029 (0X001D) RET_EMPTY_Entry
00030 (0X001E) RET_FULL_Entry
00031 (0X001F) BUF_Record_Entry
00032 (0X0020) SMH_ReportEvent0
00033 (0X0021) UNUSED_HOOK_109_033
00034 (0X0022) UNUSED_HOOK_109_034
00035 (0X0023) SSM_SmhRing0_Entry
00036 (0X0024) SSM_SmhRegister_Entry
00037 (0X0025) SSM_SmhReportEvt_Entry
00038 (0X0026) SSM_SmhNotify_Entry
00039 (0X0027) SSM_SmhDeRegister_Entry
00040 (0X0028) SSM_SmhLockMem_Entry
00041 (0X0029) SSM_SpiAssociate_Entry
00042 (0X002A) SSM_SpiCreate_Entry
00043 (0X002B) SSM_SpiDestroy_Entry
00044 (0X002C) SSM_SpiEnumProtocol_Entry
00045 (0X002D) SSM_SpiEnumHndlr_Entry
00046 (0X002E) SSM_SpiGetHndlr_Entry
00047 (0X002F) SSM_SpiGetProtocol_Entry
00048 (0X0030) SSM_SpiGetTime_Entry
00049 (0X0031) SSM_SpiInstProtocol_Entry
00050 (0X0032) SSM_SpiSeekStrm_Entry
00051 (0X0033) SSM_SpiStartStrm_Entry
00052 (0X0034) SSM_SpiStopStrm_Entry
00053 (0X0035) SSM_SpiEnableEvent_Entry
Trace Events for Multi-Media Extensions Major Code: 0X006D, Sorted by Tracepoint

ACP_ACPADevIntEntry 00080 (0X0050)
ACP_ACPADevIntExit 00208 (0X00D0)
ACP_DevReportError 00097 (0X0061)
ACP_DevReportError 00225 (0X00E1)
ADH_ADSHIntHandlerEntry 00081 (0X0051)
ADH_ADSHIntHandlerExit 00209 (0X00D1)
ADH_EventHandlerEntry 00096 (0X0060)
ADH_EventHandlerExit 00224 (0X00D0)
ADH_GetEmptyEntry 00085 (0X0055)
ADH_GetEmptyExit 00213 (0X00D5)
ADH_GetFullEntry 00087 (0X0057)
ADH_GetFullExit 00215 (0X00D7)
ADH_ReportCueDataEntry 00093 (0X005D)
ADH_ReportCueDataExit 00221 (0X00D0)
ADH_ReportCueTimeEntry 00092 (0X005C)
ADH_ReportCueTimeExit 00220 (0X00D0)
ADH_ReportDataUnderEntry 00094 (0X005E)
ADH_ReportDataUnderExit 00222 (0X00DE)
ADH_ReportEOSEntry 00099 (0X0059)
ADH_ReportEOSExit 00217 (0X00D9)
ADH_ReportErrorEntry 00090 (0X005A)
ADH_ReportErrorExit 00218 (0X00DA)
ADH_ReportPLCueEntry 00091 (0X005B)
ADH_ReportPLCueExit 00219 (0X00DB)
ADH_ReportSyncOverEntry 00095 (0X005F)
ADH_ReportSyncOverExit 00223 (0X00DF)
ADH_ReturnEmptyEntry 00088 (0X0058)
ADH_ReturnEmptyExit 00216 (0X00DB)
ADH_ReturnFullEntry 00086 (0X0056)
ADH_ReturnFullExit 00214 (0X00D6)
ADH_SHCCreateEntry 00082 (0X0052)
Multi-Media Extensions Major Code: 0X006D Minor Code: 1 (0X0001)

Description
SSM_ShcAssociate_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
1 (0X0001)

Trace Groups
SSMSRV
Multi-Media Extensions Major Code: 0X006D Minor Code: 2 (0X0002)

Description
SSM_ShcClose_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
2 (0X0002)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RingLvl=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 3 (0X0003)

Description
SSM_ShcCreate_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
3 (0X0003)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RingLvl=%F DataType=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 4 (0X0004)

Description: SSM_ShcDestroy_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 4 (0X0004)
Trace Groups: SSMSRV
Trace Types: No types assigned.
Traced Parameters:

hstream=%F hid=%F RingLvl=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 5 (0X0005)

Description: SSM_ShcStart_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 5 (0X0005)
Trace Groups: SSMSRV
Trace Types: No types assigned.
Traced Parameters:

hstream=%F hid=%F RingLvl=%F ulFlags=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 6 (0X0006)
SSM_ShcStop_Entry

Static trace point in Multi-Media Extensions.

Minor Code
6 (0X0006)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters

hstream=%F hid=%F RingLvl=%F ulFlags=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 7 (0X0007)

SSM_ShcSeek_Entry

Static trace point in Multi-Media Extensions.

Minor Code
7 (0X0007)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters

hstream=%F hid=%F RingLvl=%F ulFlags=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 8 (0X0008)

SSM_ShcEnableEvt_Entry

Static trace point in Multi-Media Extensions.

Minor Code
8 (0X0008)

Trace Groups
Multi-Media Extensions Major Code: 0X006D Minor Code: 9 (0X0009)

Description  
SSM_ShcDisableEvt_Entry

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
9 (0X0009)

Trace Groups  
SSMSRV

Trace Types  
No types assigned.

Traced Parameters

hstream=%F hid=%F RingLvl=%F hevent=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 10 (0X000A)

Description  
SSM_ShcEnableSync_Entry

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
10 (0X000A)

Trace Groups  
SSMSRV

Trace Types  
No types assigned.

Traced Parameters

hstream=%F hid=%F RingLvl=%F ulFlags=%F
<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcDisableSync_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>11 (0X000B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hid=%F RingLvl=%F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcGetTime_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>12 (0X000C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hid=%F RingLvl=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 13 (0X000D)
Description: SSM_ShcGetProtocol_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 13 (0X000D)
Trace Groups: SSMSRV
Trace Types: No types assigned.
Traced Parameters:

hid=%F RingLvl=%F DataType=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 14 (0X000E)

Description: SSM_ShcInstProtocol_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 14 (0X000E)
Trace Groups: SSMSRV
Trace Types: No types assigned.
Traced Parameters:

hid=%F RingLvl=%F ulFlags=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 15 (0X000F)

Description: SSM_ShcEnumProtocol_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 15 (0X000F)
Trace Groups:
Multi-Media Extensions Major Code: 0X006D Minor Code: 16 (0X0010)

Description
SSM_ShcNegotReslt_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
16 (0X0010)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RingLvl=%F NumKeys=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 17 (0X0011)

Description
SSM_ShcSendMsg_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
17 (0X0011)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RingLvl=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 18 (0X0012)

- **Description**: UNUSED_HOOK_109_018
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 18 (0X0012)
- **Trace Groups**: SSMSRV
- **Trace Types**: No types assigned.
- **Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 19 (0X0013)

- **Description**: UNUSED_HOOK_109_019
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 19 (0X0013)
- **Trace Groups**: SSMSRV
- **Trace Types**: No types assigned.
- **Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 20 (0X0014)
SSM_ProcRun_Entry

Description
Static trace point in Multi-Media Extensions.

Minor Code
20 (0X0014)

Trace Groups
SSMDD

Trace Types
No types assigned.

Traced Parameters
ProcKey=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 21 (0X0015)

SSM_ProcBlock_Entry

Description
Static trace point in Multi-Media Extensions.

Minor Code
21 (0X0015)

Trace Groups
SSMDD

Trace Types
No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 22 (0X0016)

SSM_IOCTL_Entry

Description
Static trace point in Multi-Media Extensions.

Minor Code
22 (0X0016)

Trace Groups
SSMDD

**Trace Types**
No types assigned.

**Traced Parameters**
Funct/Catg=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 23 (0X0017)

**Description**
SSM_IDC_Call_To_SHC

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
23 (0X0017)

**Trace Groups**
SSMDD

**Trace Types**
No types assigned.

**Traced Parameters**
hid=%F Function=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 24 (0X0018)

**Description**
SSM_IDCNotifyEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
24 (0X0018)

**Trace Groups**
SSMDD

**Trace Types**
No types assigned.

**Traced Parameters**
hidstream=%F htid=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 25 (0X0019)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_IDCDeRegEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>25 (0X0019)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMDD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 26 (0X001A)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_IDCRptEvtEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>26 (0X001A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMDD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

hid=%F ulType=%F ulSubType=%F hstream=%F ulStatus=%F LowParm1=%F HighParm1=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 27 (0X001B)
**Description**

GET_EMPTY_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

27 (0X001B)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hstreamOwner=%F hbcbFull=%F hbcbEmpty=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 28 (0X001C)

**Description**

GET_FULL_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

28 (0X001C)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hstreamOwner=%F hbcbFull=%F hbcbEmpty=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 29 (0X001D)

**Description**

RET_EMPTY_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

29 (0X001D)

**Trace Groups**
SSMN/TIC

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{hstream}=%F \quad \text{pBuffer}=%F \quad \text{pRecord}=%F \quad \text{ulLength}=%F \quad \text{ulBDSFlag}=%F \quad \text{ulSCBFlag}=%F \quad \text{ulNumFull}=%F
\]

Multi-Media Extensions Major Code: 0X006D Minor Code: 30 (0X001E)

**Description**
RET_FULL_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
30 (0X001E)

**Trace Groups**
SSMN/TIC

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{ulNumUsers}=%F \quad \text{ulNumRecords}=%F \quad \text{rcbHead}=%F \quad \text{rcbFull}=%F \quad \text{NumAllFull}=%F \quad \text{NumSrcOwn}=%F \quad \text{NumTgtOwn}=%F
\]

Multi-Media Extensions Major Code: 0X006D Minor Code: 31 (0X001F)

**Description**
BUF_Record_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
31 (0X001F)

**Trace Groups**
SSMN/TIC

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{ulNumUsers}=%F \quad \text{ulNumRecords}=%F \quad \text{rcbHead}=%F \quad \text{rcbFull}=%F \quad \text{NumAllFull}=%F \quad \text{NumSrcOwn}=%F \quad \text{NumTgtOwn}=%F
\]
Multi-Media Extensions Major Code: 0X006D Minor Code: 32 (0X0020)

**Description**  
SMH_ReportEvent0

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
32 (0X0020)

**Trace Groups**  
SSMEVDDC

**Trace Types**  
No types assigned.

**Traced Parameters**

hstream=%F hid=%F ulType=%F ulSubType=%F ulStatus=%F ulParm1=%F ulParm2=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 33 (0X0021)

**Description**  
UNUSED_HOOK_109_033

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
33 (0X0021)

**Trace Groups**  
SSMEVDDC

**Trace Types**  
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 34 (0X0022)
<table>
<thead>
<tr>
<th>Description</th>
<th>UNUSED_HOOK_109_034</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>34 (0X0022)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMEVDDC</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 35 (0X0023)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SmhRing3_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>35 (0X0023)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Function#=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 36 (0X0024)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SmhRegister_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>36 (0X0024)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 37 (0X0025)

**Description**  
SSM_SmhReportEvnt_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
37 (0X0025)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**  

- hstream=%F
- hevent=%F
- hid=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 38 (0X0026)

**Description**  
SSM_SmhNotify_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
38 (0X0026)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**  

- hstream=%F
- hid=%F
- Flags=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 39 (0X0027)

Description: SSM_SmhDeRegister_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 39 (0X0027)
Trace Groups: SSMAPI
Trace Types: No types assigned.
Traced Parameters: Function#=%F

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 40 (0X0028)

Description: SSM_SmhLockMem_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 40 (0X0028)
Trace Groups: SSMAPI
Trace Types: No types assigned.
Traced Parameters: Function#=%F Flags=%F

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 41 (0X0029)
Description  SSM_SpiAssociate_Entry
Tracepoint  Static trace point in Multi-Media Extensions.
Minor Code  41 (0X0029)
Trace Groups  SSMAPI
Trace Types  No types assigned.
Traced Parameters

hstream=%F hid=%F

--------------------------------------------
Multi-Media Extensions Major Code: 0X006D Minor Code: 42 (0X002A)

Description  SSM_SpiCreate_Entry
Tracepoint  Static trace point in Multi-Media Extensions.
Minor Code  42 (0X002A)
Trace Groups  SSMAPI
Trace Types  No types assigned.
Traced Parameters

hstreamBuf=%F hidSrc=%F hidTgt=%F DataType=%F

--------------------------------------------
Multi-Media Extensions Major Code: 0X006D Minor Code: 43 (0X002B)

Description  SSM_SpiDestroy_Entry
Tracepoint  Static trace point in Multi-Media Extensions.
Minor Code  43 (0X002B)
Trace Groups
Multi-Media Extensions Major Code: 0X006D Minor Code: 44 (0X002C)

**Description**
SSM_SpiEnumProtocol_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
44 (0X002C)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 45 (0X002D)

**Description**
SSM_SpiEnumHndlr_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
45 (0X002D)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F

-------------------------------
Multi-Media Extensions Major Code: 0X006D Minor Code: 46 (0X002E)

Description
SSM_SpiGetHndlr_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
46 (0X002E)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
N=%F A=%F M=%F E=%F !=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 47 (0X002F)

Description
SSM_SpiGetProtocol_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
47 (0X002F)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hid=%F Datatype=%F DataSubType=%F ullntKey=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 48 (0X0030)
SSM_SpiGetTime_Entry

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
48 (0X0030)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

```
hstream=%F
```

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 49 (0X0031)

SSM_SpiInstProtocol_Entry

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
49 (0X0031)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

```
hid=%F Datatype=%F DataSubType=%F ulIntKey=%F ulFlags=%F
```

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 50 (0X0032)

SSM_SpiSeekStrm_Entry

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
50 (0X0032)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F Flags=%F lSeekH=%F lSeekL=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 51 (0X0033)

**Description**
SSM_SpiStartStrm_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
51 (0X0033)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F Flags=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 52 (0X0034)

**Description**
SSM_SpiStopStrm_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
52 (0X0034)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F Flags=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 53 (0X0035)

**Description**  
SSM_SpiEnableEvent_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
53 (0X0035)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hstream=%F hid=%F uiType=%F ulSubType=%F Parm1=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 54 (0X0036)

**Description**  
SSM_SpiDisableEvent_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
54 (0X0036)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hevent=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 55 (0X0037)
**SSM_SpiEnableSync_Entry**

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
55 (0X0037)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{hstreamMast} = \%F \quad \text{slaves} = \%F \quad \text{mmtimeSync} = \%F
\]

---

**Multi-Media Extensions Major Code: 0X006D Minor Code: 56 (0X0038)**

**SSM_SpiDisableSync_Entry**

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
56 (0X0038)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

\[
hstream = \%F
\]

---

**Multi-Media Extensions Major Code: 0X006D Minor Code: 57 (0X0039)**

**SSM_ShcRouter_Entry**

**Description**
Static trace point in Multi-Media Extensions.

**Minor Code**
57 (0X0039)

**Trace Groups**
Multi-Media Extensions Major Code: 0X006D Minor Code: 58 (0X003A)

Description  SSM_ShcRouterRing3_Entry
Tracepoint  Static trace point in Multi-Media Extensions.
Minor Code  58 (0X003A)
Trace Groups  SSMSRV
Trace Types  No types assigned.
Traced Parameters  

hid=%F

--------------------------------------------
Multi-Media Extensions Major Code: 0X006D Minor Code: 59 (0X003B)

Description  SSM_SmhRptEventSync_Entry
Tracepoint  Static trace point in Multi-Media Extensions.
Minor Code  59 (0X003B)
Trace Groups  SSMEVDLL
Trace Types  No types assigned.
Traced Parameters  

hstreamSlave=%F mtimeSlave=%F hstreamMast=%F hidMast=%F mmtimeMast=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 60 (0X003C)

- **Description**: SSM_AppEvent3_Entry
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 60 (0X003C)
- **Trace Groups**: SSMEVDLL
- **Trace Types**: No types assigned.
- **Traced Parameters**: hstream=%F hevent=%F ulType=%F ulSubType=%F ulStatus=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 61 (0X003D)

- **Description**: SSM_AppEventQ_Entry
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 61 (0X003D)
- **Trace Groups**: SSMEVDLL
- **Trace Types**: No types assigned.
- **Traced Parameters**: hstream=%F hevent=%F ulType=%F ulSubType=%F ulStatus=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 62 (0X003E)
SSM_SmhNotifyIOCtl_Entry

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 62 (0X003E)

Trace Groups: SSMSRV

Trace Types: No types assigned.

Traced Parameters:

hstream=%F hid=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 63 (0X003F)

SSM_QueueEventIOCtl_Entry

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 63 (0X003F)

Trace Groups: SSMSRV

Trace Types: No types assigned.

Traced Parameters:

hid=%F ulType=%F ulSubType=%F hstream=%F ulStatus=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 64 (0X0040)

SSM_CreateBDS

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 64 (0X0040)

Trace Groups:
Multi-Media Extensions Major Code: 0X006D Minor Code: 65 (0X0041)

Description
SSM_SpiDetSyncMaster_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
65 (0X0041)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters

#masters=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 66 (0X0042)

Description
SSM_SpiSendMsg_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
66 (0X0042)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters

hstream=%F hid=%F ulMsgType=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 67 (0X0043)

**Description**
SSM_SpiAssStrNetEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
67 (0X0043)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F ulFlags=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 68 (0X0044)

**Description**
SSM_SpiCloseStrNetEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
68 (0X0044)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 69 (0X0045)
SSM_SpiConnStrEntry

Static trace point in Multi-Media Extensions.

Minor Code
69 (0X0045)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters

hidSrc=%F hidTgt=%F datatype=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 70 (0X0046)

SSM_SpiDeactStrEntry

Static trace point in Multi-Media Extensions.

Minor Code
70 (0X0046)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters

hstream=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 71 (0X0047)

SSM_SpiAddStrConnEntry

Static trace point in Multi-Media Extensions.

Minor Code
71 (0X0047)

Trace Groups
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F hstream=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 72 (0X0048)

**Description**
SSM_SpiOpenStrNetEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
72 (0X0048)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 73 (0X0049)

**Description**
SSM_SpiReactStrEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
73 (0X0049)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F
### Multi-Media Extensions Major Code: 0X006D Minor Code: 74 (0X004A)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SpiRemStrConnEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>74 (0X004A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 75 (0X004B)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SpiSeekStrNetEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>75 (0X004B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hnetwork=%F ulFlags=%F iSeekPoint=%F</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 76 (0X004C)
<table>
<thead>
<tr>
<th>Major Code</th>
<th>Minor Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X006D</td>
<td>0X004C</td>
</tr>
<tr>
<td>0X006D</td>
<td>0X004D</td>
</tr>
<tr>
<td>0X006D</td>
<td>0X004E</td>
</tr>
</tbody>
</table>

**SSM_SpiStartStrNetEntry**

**Description**: Static trace point in Multi-Media Extensions.

**Minor Code**: 76 (0X004C)

**Trace Groups**: SSMAPI

**Trace Types**: No types assigned.

**Traced Parameters**

```

hnetwork=%F ulFlags=%F

```

**SSM_SpiStopStrNetEntry**

**Description**: Static trace point in Multi-Media Extensions.

**Minor Code**: 77 (0X004D)

**Trace Groups**: SSMAPI

**Trace Types**: No types assigned.

**Traced Parameters**

```

hnetwork=%F ulFlags=%F

```

**SSM_SpiUnassStrNetEntry**

**Description**: Static trace point in Multi-Media Extensions.

**Minor Code**: 78 (0X004E)

**Trace Groups**: SSMAPI

**Trace Types**: No types assigned.

**Traced Parameters**

```

hnetwork=%F ulFlags=%F

```
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F

-----------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 80 (0X0050)

**Description**
ACP_ACPADevintEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
80 (0X0050)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

Card=%F ID=%F

-----------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 81 (0X0051)

**Description**
ADH_ADSHIntHandlerEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
81 (0X0051)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

hStream=%F pBuffer=%F ulFlag=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 82 (0X0052)

**Description**
ADH_SHCCreateEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
82 (0X0052)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 83 (0X0053)

**Description**
ADH_SHCStartEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
83 (0X0053)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F ulFlags=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 84 (0X0054)
Description: ADH_SHCStopEntry

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 84 (0x0054)

Trace Groups: ADSH

Trace Types: No types assigned.

Traced Parameters:

  hstream=%F ulFlags=%F

Multi-Media Extensions Major Code: 0x006D Minor Code: 85 (0x0055)

Description: ADH_GetEmptyEntry

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 85 (0x0055)

Trace Groups: ADSH

Trace Types: No types assigned.

Traced Parameters:

  hStream=%F RC=%F pBuffer=%F ulFlags=%F

Multi-Media Extensions Major Code: 0x006D Minor Code: 86 (0x0056)

Description: ADH_ReturnFullEntry

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 86 (0x0056)

Trace Groups:
ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

--------------------------------------------

**Multi-Media Extensions Major Code: 0X006D Minor Code: 87 (0X0057)**

**Description**

ADH_GetFullEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

87 (0X0057)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

--------------------------------------------

**Multi-Media Extensions Major Code: 0X006D Minor Code: 88 (0X0058)**

**Description**

ADH_ReturnEmptyEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

88 (0X0058)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F
| Multi-Media Extensions Major Code: 0X006D Minor Code: 89 (0X0059) |
|---|---|---|---|
| **Description** | ADH_ReportEOSEntry |
| **Tracepoint** | Static trace point in Multi-Media Extensions. |
| **Minor Code** | 89 (0X0059) |
| **Trace Groups** | ADSH |
| **Trace Types** | No types assigned. |
| **Traced Parameters** | hStream=%F RC=%F pBuffer=%F ulFlags=%F |

| Multi-Media Extensions Major Code: 0X006D Minor Code: 90 (0X005A) |
|---|---|---|---|
| **Description** | ADH_ReportErrorEntry |
| **Tracepoint** | Static trace point in Multi-Media Extensions. |
| **Minor Code** | 90 (0X005A) |
| **Trace Groups** | ADSH |
| **Trace Types** | No types assigned. |
| **Traced Parameters** | No parameters traced. |

| Multi-Media Extensions Major Code: 0X006D Minor Code: 91 (0X005B) |
|---|---|---|---|
| **Description** |  |  |  |
ADH_ReportPlCueEntry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
91 (0X005B)

**Trace Groups**  
ADSH

**Trace Types**  
No types assigned.

**Traced Parameters**  
No parameters traced.

---------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 92 (0X005C)

**Description**  
ADH_ReportCueTimeEntry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
92 (0X005C)

**Trace Groups**  
ADSH

**Trace Types**  
No types assigned.

**Traced Parameters**  
No parameters traced.

---------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 93 (0X005D)

**Description**  
ADH_ReportCueDataEntry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
93 (0X005D)

**Trace Groups**  
ADSH

**Trace Types**  
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 94 (0X005E)

Description
ADH_ReportDataUnderEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
94 (0X005E)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters

hStream=%F ulStatus=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 95 (0X005F)

Description
ADH_ReportSyncOverEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
95 (0X005F)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters

hStream=%F ulStatus=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 96
**ADH_EventHandlerEntry**

**Description**
ADH_EventHandlerEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
96 (0X0060)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

hStream=%F ulStreamTime=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 97 (0X0061)

**Description**
ACP_DevReportError

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
97 (0X0061)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

Error(DSP/MME)=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 101 (0X0065)

**Description**
ADSH_AddBufEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

Minor Code 101 (0X0065)
Trace Groups ADSH
Trace Types No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 102 (0X0066)

Description ADSH_RemoveBufEntry
Tracepoint Static trace point in Multi-Media Extensions.
Minor Code 102 (0X0066)
Trace Groups ADSH
Trace Types No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 103 (0X0067)

Description ADSH_IDCWriteEntry
Tracepoint Static trace point in Multi-Media Extensions.
Minor Code 103 (0X0067)
Trace Groups ADSH
Trace Types No types assigned.
Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 104 (0X0068)

Description
ADSH_GetBufferEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
104 (0X0068)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 105 (0X0069)

Description
ADSH_RetBufferEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
105 (0X0069)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code:
### 106 (0X006A)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>ADSH_DLLIntEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>106 (0X006A)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>ADSH</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 107 (0X006B)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>ADSH_IDCControlEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>107 (0X006B)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>ADSH</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 108 (0X006C)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>ADSH_IDCSetupEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td></td>
</tr>
</tbody>
</table>

---
Static trace point in Multi-Media Extensions.

**Minor Code**
108 (0X006C)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 109 (0X006D)

**Description**
ADSH_SendBufferEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
109 (0X006D)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 112 (0X0070)

**Description**
CDSH_StartCdReadIoctl

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
112 (0X0070)

**Trace Groups**
CD

**Trace Types**
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 113 (0X0071)

Description
CDSH_StartCddaRead

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
113 (0X0071)

Trace Groups
CD

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 114 (0X0072)

Description
CDSH_StartCddaWrite

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
114 (0X0072)

Trace Groups
CD

Trace Types
No types assigned.

Traced Parameters
129 (0X0081)

**Description**
SSM_ShcAssociate_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
129 (0X0081)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 130 (0X0082)

**Description**
SSM_ShcClose_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
130 (0X0082)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 131 (0X0083)

**Description**
SSM_ShcCreate_Exit

**Tracepoint**
### Multi-Media Extensions Major Code: 0X006D Minor Code: 131 (0X0083)

<table>
<thead>
<tr>
<th>Description</th>
<th>Static trace point in Multi-Media Extensions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>131 (0X0083)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hid=%F RC=%F</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 132 (0X0084)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcDestroy.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>132 (0X0084)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hid=%F RC=%F</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 133 (0X0085)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcStart.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>133 (0X0085)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hid=%F RC=%F</td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 134 (0X0086)

Description
SSM_ShcStop_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
134 (0X0086)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 135 (0X0087)

Description
SSM_ShcSeek_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
135 (0X0087)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code:
136 (0X0088)

Description  
SSM_ShcEnableEvt_Exit

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
136 (0X0088)

Trace Groups  
SSMSRV

Trace Types  
No types assigned.

Traced Parameters

hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 137 (0X0089)

Description  
SSM_ShcDisableEvt_Exit

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
137 (0X0089)

Trace Groups  
SSMSRV

Trace Types  
No types assigned.

Traced Parameters

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 138 (0X008A)

Description  
SSM_ShcEnableSync_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

**Minor Code**
138 (0X008A)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 139 (0X008B)

**Description**
SSM_ShcDisableSync_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
139 (0X008B)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 140 (0X008C)

**Description**
SSM_ShcGetTime_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
140 (0X008C)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.
Traced Parameters

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 141 (0X008D)

Description
SSM_ShcGetProtocol_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
141 (0X008D)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters

hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 142 (0X008E)

Description
SSM_ShcInstProtocol_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
142 (0X008E)

Trace Groups
SSMSRV

Trace Types
No types assigned.

Traced Parameters

hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code:
143 (0X008F)

**Description**
SSM_ShcEnumProtocol_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
143 (0X008F)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**
hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 144 (0X0090)

**Description**
SSM_ShcNegotReslt_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
144 (0X0090)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 145 (0X0091)

**Description**
SSM_ShcSendMsg_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
145 (0X0091)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 146 (0X0092)

**Description**
UNUSED_HOOK_109_146

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
146 (0X0092)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 147 (0X0093)

**Description**
UNUSED_HOOK_109_147

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
147 (0X0093)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.
## Traced Parameters

### Multi-Media Extensions Major Code: 0X006D Minor Code: 148 (0X0094)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ProcRun.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>148 (0X0094)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMDD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Traced Parameters

- UnblockFlag=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 149 (0X0095)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ProcBlock.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>149 (0X0095)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMDD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Traced Parameters

- UnblockFlag=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code:
150 (0X0096)

Description
SSM_ioctl Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
150 (0X0096)

Trace Groups
SSMDD

Trace Types
No types assigned.

Traced Parameters
Funct/Catg=%F RC=%F

---------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 151 (0X0097)

Description
SSM_IDC_Ret_From_SHC

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
151 (0X0097)

Trace Groups
SSMDD

Trace Types
No types assigned.

Traced Parameters
hid=%F Function=%F RC=%F

---------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 152 (0X0098)

Description
SSM_IDCNotifyExit

Tracepoint
Static trace point in Multi-Media Extensions.

**Minor Code**
152 (0X0098)

**Trace Groups**
SSMDD

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 153 (0X0099)

**Description**
SSM_IDCDeRegExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
153 (0X0099)

**Trace Groups**
SSMDD

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 154 (0X009A)

**Description**
SSM_IDCRptEvtExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
154 (0X009A)

**Trace Groups**
SSMDD

**Trace Types**
No types assigned.
Traced Parameters

hid=%F RC=%F LowParm2=%F HighParm2=%F LowParm3=%F HighParm3=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 155 (0X009B)

Description
GET_EMPTY_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
155 (0X009B)

Trace Groups
SSMNOTIC

Trace Types
No types assigned.

Traced Parameters

hstream=%F pBufferP=%F pBuffer=%F pRecord=%F ulLength=%F hbcbFull=%F hbcbEmpty=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 156 (0X009C)

Description
GET_FULL.Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
156 (0X009C)

Trace Groups
SSMNOTIC

Trace Types
No types assigned.

Traced Parameters

hstream=%F pBufferP=%F pBuffer=%F pRecord=%F ulLength=%F hbcbFull=%F hbcbEmpty=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code:
### 157 (0X009D)

<table>
<thead>
<tr>
<th>Description</th>
<th>RET_EMPTY_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>157 (0X009D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMNOTIC</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hstreamOwner=%F pBufferP=%F hbcbThis=%F hbcbFull=%F hbcbEmpty=%F</td>
</tr>
</tbody>
</table>

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 158 (0X009E)

<table>
<thead>
<tr>
<th>Description</th>
<th>RET_FULL_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>158 (0X009E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMNOTIC</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hstreamOwner=%F pBufferP=%F hbcbThis=%F hbcbFull=%F hbcbEmpty=%F rc=%F</td>
</tr>
</tbody>
</table>

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 159 (0X009F)

<table>
<thead>
<tr>
<th>Description</th>
<th>BUF_Record_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td></td>
</tr>
</tbody>
</table>
Static trace point in Multi-Media Extensions.

**Minor Code**
159 (0X009F)

**Trace Groups**
SSMNOTIC

**Trace Types**
No types assigned.

**Traced Parameters**
ulNumUsers=%F ulNumRecords=%F rcbHead=%F rcbFull=%F NumAllFull=%F NumSrcOwn=%F NumATgtOwn=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 160 (0X00A0)

**Description**
SSM_Negotiate_error

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
160 (0X00A0)

**Trace Groups**
SSMEVDDC

**Trace Types**
No types assigned.

**Traced Parameters**
hstream=%F ErrorStatus=%F KeyDataType=%F KeySubType=%F KeyIntKey=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 161 (0X00A1)

**Description**
UNUSED_HOOK_109_161

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
161 (0X00A1)

**Trace Groups**
SSMEVDDC

**Trace Types**
Traced Parameters

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 162 (0X00A2)

Description  UNUSED_HOOK_109_162
Tracepoint   Static trace point in Multi-Media Extensions.
Minor Code   162 (0X00A2)
Trace Groups SSMEVDDC
Trace Types  No types assigned.
Traced Parameters

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 163 (0X00A3)

Description  SSM_SmhRing3_Exit
Tracepoint   Static trace point in Multi-Media Extensions.
Minor Code   163 (0X00A3)
Trace Groups SSMAPI
Trace Types  No types assigned.
Traced Parameters

    Function#=%F RC=%F

--------------------------------------------
Multi-Media Extensions Major Code: 0X006D Minor Code: 164 (0X00A4)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SmhRegister_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>164 (0X00A4)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hidSrc=%F hidTgt=%F</td>
</tr>
</tbody>
</table>

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 165 (0X00A5)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_SmhReportEvnt_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>165 (0X00A5)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hstream=%F hevent=%F RC=%F</td>
</tr>
</tbody>
</table>

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 166 (0X00A6)

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>

SSM_SmhNotify_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
166 (0X00A6)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 167 (0X00A7)

Description
SSM_SmhDeRegister_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
167 (0X00A7)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
Function#=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 168 (0X00A8)

Description
SSM_SmhLockMem_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
168 (0X00A8)

Trace Groups
SSMAPI
Multi-Media Extensions Major Code: 0X006D Minor Code: 169 (0X00A9)

Description
SSM_SpiAssociate_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
169 (0X00A9)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstream=%F hid=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 170 (0X00AA)

Description
SSM_SpiCreate_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
170 (0X00AA)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstreamBuf=%F hstream=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 171 (0X00AB)

**Description**  
SSM_SpiDestroy_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
171 (0X00AB)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hstream=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 172 (0X00AC)

**Description**  
SSM_SpiEnumProtocol_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
172 (0X00AC)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hid=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 173 (0X00AD)
SSM_SpiEnumHndlr_Exit

Static trace point in Multi-Media Extensions.

Minor Code 173 (0X00AD)

Trace Groups SSMAPI

Trace Types No types assigned.

Traced Parameters

#Handlers=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 174 (0X00AE)

SSM_SpiGetHndlr_Exit

Static trace point in Multi-Media Extensions.

Minor Code 174 (0X00AE)

Trace Groups SSMAPI

Trace Types No types assigned.

Traced Parameters

hidSrc=%F hidTgt=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 175 (0X00AF)

SSM_SpiGetProtocol_Exit

Static trace point in Multi-Media Extensions.

Minor Code 175 (0X00AF)

Trace Groups
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F Datatype=%F RC=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 176 (0X00B0)

**Description**
SSM_SpiGetTime_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
176 (0X00B0)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F mmtimeH=%F mmtimeL=%F RC=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 177 (0X00B1)

**Description**
SSM_SpiInstProtocol_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
177 (0X00B1)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F Datatype=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 178 (0X00B2)

**Description**
SSM_SpiSeekStrm_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
178 (0X00B2)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 179 (0X00B3)

**Description**
SSM_SpiStartStrm_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
179 (0X00B3)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 180 (0X00B4)
SSM_SpiStopStrm_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
180 (0X00B4)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstream=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 181 (0X00B5)

SSM_SpiEnableEvent_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
181 (0X00B5)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstream=%F hevent=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 182 (0X00B6)

SSM_SpiDisableEvent_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
182 (0X00B6)

Trace Groups

Multi-Media Extensions Major Code: 0X006D Minor Code: 183 (0X00B7)

Description
SSM_SpiEnableSync_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
183 (0X00B7)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstreamMast=%F #slaves=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 184 (0X00B8)

Description
SSM_SpiDisableSync_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
184 (0X00B8)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hstream=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 185 (0X00B9)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcRouter_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>185 (0X00B9)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hid=%F RC=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 186 (0X00BA)

<table>
<thead>
<tr>
<th>Description</th>
<th>SSM_ShcRouterRing3_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>186 (0X00BA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SSMSRV</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hid=%F RC=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 187 (0X00BB)
## Multi-Media Extensions Major Code: 0X006D Minor Code: 187 (0X00BB)

- **Description**: SSM_xxxUnused
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 187 (0X00BB)
- **Trace Groups**: SSMEVDLL
- **Trace Types**: No types assigned.

### Traced Parameters

- hstream=%F hevent=%F hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 188 (0X00BC)

- **Description**: SSM_AppEvent3_Exits
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 188 (0X00BC)
- **Trace Groups**: SSMEVDLL
- **Trace Types**: No types assigned.

### Traced Parameters

- hstream=%F hevent=%F hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 189 (0X00BD)

- **Description**: SSM_AppEventQ_Exits
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 189 (0X00BD)
- **Trace Groups**: SSMEVDLL
- **Trace Types**: No types assigned.

### Traced Parameters

- hstream=%F hevent=%F hid=%F
SSMEVDLL

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hevent=%F hid=%F

--------------------------------------------

**Description**
SSM_SmhNotifyIOCtl_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
190 (0X00BE)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid0/RC3=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F RC0=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 190 (0X00BE)

**Description**

SSM_SmhNotifyIOCtl_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
190 (0X00BE)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F hid0/RC3=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F RC0=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 191 (0X00BF)

**Description**

SSM_QueueEventIOCtl_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
191 (0X00BF)

**Trace Groups**
SSMSRV

**Trace Types**
No types assigned.

**Traced Parameters**

hid=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 192 (0X00C0)

Description: SSM_CreateBDSNotMax
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 192 (0X00C0)
Trace Groups: SSMBUFF
Trace Types: No types assigned.
Traced Parameters:

ulMinBuf=%F ulMaxBuf=%F ulNumBCB=%F ulHeapSize=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 193 (0X00C1)

Description: SSM_SpiDetSyncMaster.Exit
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 193 (0X00C1)
Trace Groups: SSMAPI
Trace Types: No types assigned.
Traced Parameters:

#masters=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 194 (0X00C2)
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Traced Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>194 (0x00C2)</td>
<td>hstream=%F hid=%F RC=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0x006D Minor Code: 195 (0x00C3)

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Traced Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>196 (0x00C4)</td>
<td>hnetwork=%F RC=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0x006D Minor Code: 196 (0x00C4)
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 197 (0X00C5)

**Description**
SSM_SpiConnStrExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
197 (0X00C5)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hidSrc=%F hidTgt=%F hstream=%F RC=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 198 (0X00C6)

**Description**
SSM_SpiDeactStrExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
198 (0X00C6)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hstream=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 199 (0X00C7)

**Description**  
SSM_SpiAddStrConnExit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
199 (0X00C7)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hnetwork=%F hstream=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 200 (0X00C8)

**Description**  
SSM_SpiOpenStrNetExit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
200 (0X00C8)

**Trace Groups**  
SSMAPI

**Trace Types**  
No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 201 (0X00C9)
**Description**  SSM_SpiReactStrExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  201 (0X00C9)

**Trace Groups**  SSMAPI

**Trace Types**  No types assigned.

**Traced Parameters**

```
hsream=%F RC=%F
```

----------------------------------------------------------------------------------

**Multi-Media Extensions Major Code: 0X006D Minor Code: 202 (0X00CA)**

**Description**  SSM_SpiRemStrConnExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  202 (0X00CA)

**Trace Groups**  SSMAPI

**Trace Types**  No types assigned.

**Traced Parameters**

```
hsream=%F RC=%F
```

----------------------------------------------------------------------------------

**Multi-Media Extensions Major Code: 0X006D Minor Code: 203 (0X00CB)**

**Description**  SSM_SpiSeekStrNetExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  203 (0X00CB)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

--------------------------------------------

### Multi-Media Extensions

**Major Code:** 0X006D  **Minor Code:** 204 (0X00CC)

**Description**
SSM_SpiStartStrNetExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
204 (0X00CC)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

--------------------------------------------

### Multi-Media Extensions

**Major Code:** 0X006D  **Minor Code:** 205 (0X00CD)

**Description**
SSM_SpiStopStrNetExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
205 (0X00CD)

**Trace Groups**
SSMAPI

**Trace Types**
No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 206 (0X00CE)

Description
SSM_SpiUnassStrNetExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
206 (0X00CE)

Trace Groups
SSMAPI

Trace Types
No types assigned.

Traced Parameters
hnetwork=%F RC=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 208 (0X00D0)

Description
ACP_ACPADevIntExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
208 (0X00D0)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters
Card=%F ID=%F

-------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 209 (0X00D1)
<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_ADSHIntHandlerExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>209 (0X00D1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F RC=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 210 (0X00D2)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_SHCCreateExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>210 (0X00D2)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 211 (0X00D3)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_SHCStartExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>211 (0X00D3)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 212 (0X00D4)

Description
ADH_SHCStopExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
212 (0X00D4)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters
hStream=%F RC=%F ulStateFlg=%F

------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 213 (0X00D5)

Description
ADH_GetEmptyExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
213 (0X00D5)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters
hStream=%F RC=%F pBuffer=%F ulFlags=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 214 (0X00D6)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_ReturnFullExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>214 (0X00D6)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F RC=%F pBuffer=%F ulFlags=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 215 (0X00D7)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_GetFullExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>215 (0X00D7)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F RC=%F pBuffer=%F ulFlags=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 216 (0X00D8)
<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_ReturnEmptyExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>216 (0X00D8)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F RC=%F pBuffer=%F ulFlags=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 217 (0X00D9)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_ReportEOSExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>217 (0X00D9)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hStream=%F RC=%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 218 (0X00DA)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADH_ReportErrorExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>218 (0X00DA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
<tr>
<td>Trace Types</td>
<td></td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 219 (0X00DB)

Description
ADH_ReportPLCueExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
219 (0X00DB)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 220 (0X00DC)

Description
ADH_ReportCueTimeExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
220 (0X00DC)

Trace Groups
ADSH

Trace Types
No types assigned.

Traced Parameters
No parameters traced.
221 (0X00DD)

**Description**
ADH_ReportCueDataExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
221 (0X00DD)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 222 (0X00DE)

**Description**
ADH_ReportDataUnderExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
222 (0X00DE)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

hStream=%F ulStatus=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 223 (0X00DF)

**Description**
ADH_ReportSyncOverExit

**Tracepoint**
Static trace point in Multi-Media Extensions.
Minor Code: 223 (0X00DF)
Trace Groups: ADSH
Trace Types: No types assigned.
Traced Parameters:

hStream=%F ulStatus=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 224 (0X00E0)

Description: ADH_EventHandlerExit
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 224 (0X00E0)
Trace Groups: ADSH
Trace Types: No types assigned.
Traced Parameters:

hStream=%F ulStreamTime=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 225 (0X00E1)

Description: ACP_DevReportError
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 225 (0X00E1)
Trace Groups: ADSH
Trace Types: No types assigned.
Traced Parameters:
Multi-Media Extensions Major Code: 0X006D Minor Code: 229 (0X00E5)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADSH_AddBufExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>229 (0X00E5)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 230 (0X00E6)

<table>
<thead>
<tr>
<th>Description</th>
<th>ADSH_RemoveBufExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>230 (0X00E6)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code:
231 (0X00E7)

**Description**
ADSH_IDCWriteExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
231 (0X00E7)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

-----------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 232 (0X00E8)

**Description**
ADSH_GetBufferExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
232 (0X00E8)

**Trace Groups**
ADSH

**Trace Types**
No types assigned.

**Traced Parameters**

-----------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 233 (0X00E9)

**Description**
ADSH_RetBufferExit

**Tracepoint**
<table>
<thead>
<tr>
<th>Description</th>
<th>ADSH_DLLIntExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>233 (0X00E9)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>ADSH_IDCControlExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>234 (0X00EA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>ADSH_IDCControlExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>235 (0X00EB)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>ADSH</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
Traced Parameters

-----------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 236 (0X00EC)

Description       ADSH_IDCSetupExit
Tracepoint         Static trace point in Multi-Media Extensions.
Minor Code         236 (0X00EC)
Trace Groups       ADSH
Trace Types        No types assigned.
Traced Parameters

-----------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 237 (0X00ED)

Description       ADSH_SendBufferExit
Tracepoint         Static trace point in Multi-Media Extensions.
Minor Code         237 (0X00ED)
Trace Groups       ADSH
Trace Types        No types assigned.
Traced Parameters

-----------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code:
### 240 (0X00F0)

<table>
<thead>
<tr>
<th>Description</th>
<th>CDSH_StopCdReadIoctl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>240 (0X00F0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>CD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 241 (0X00F1)

<table>
<thead>
<tr>
<th>Description</th>
<th>CDSH_StopCddaRead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>241 (0X00F1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>CD</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 242 (0X00F2)

<table>
<thead>
<tr>
<th>Description</th>
<th>CDSH_StopCddaWrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td></td>
</tr>
</tbody>
</table>
Static trace point in Multi-Media Extensions.

**Minor Code**
242 (0X00F2)

**Trace Groups**
CD

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 257 (0X0101)

**Description**
MDM_mciSendStringEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
257 (0X0101)

**Trace Groups**
MMSNDSTR

**Trace Types**
No types assigned.

**Traced Parameters**

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 258 (0X0102)

**Description**
MDM_mciSendCommandEntry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
258 (0X0102)

**Trace Groups**
MMSNDCMD

**Trace Types**
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 259 (0X0103)

Description
MDM_mciOpen_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
259 (0X0103)

Trace Groups
MCIOPEN

Trace Types
No types assigned.

Traced Parameters
UserParm=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 260 (0X0104)

Description
MDM_MCDOpen_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
260 (0X0104)

Trace Groups
MCIOPEN

Trace Types
No types assigned.

Traced Parameters
UserParm=%F
261 (0X0105)

**Description**
MDM_mciMakeActive_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
261 (0X0105)

**Trace Groups**
MCIOPEN

**Trace Types**
No types assigned.

**Traced Parameters**
ID=%F Type=%F RU=%F Class=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 262 (0X0106)

**Description**
MDM_mciClose_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
262 (0X0106)

**Trace Groups**
MCICLOSE

**Trace Types**
No types assigned.

**Traced Parameters**
ID=%F UserParm=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 263 (0X0107)

**Description**
MDM_RESTORE_Entry

**Tracepoint**
Multi-Media Extensions Major Code: 0X006D Minor Code: 263 (0X0107)

Description
MDM_SAVE_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
263 (0X0107)

Trace Groups
MMRESMAN

Trace Types
No types assigned.

Traced Parameters
ID=%F Type=%F RU=%F Class=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 264 (0X0108)

Description
MDM_SAVE_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
264 (0X0108)

Trace Groups
MMRESMAN

Trace Types
No types assigned.

Traced Parameters
ID=%F Type=%F RU=%F Class=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 265 (0X0109)

Description
MDM_Thread_call_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
265 (0X0109)

Trace Groups
MMRESMAN

Trace Types
No types assigned.
Traced Parameters

ID=%F Message=%F UserParm=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 324 (0X0144)

Description
MTSH_GetEmptyEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
324 (0X0144)

Trace Groups
MTSHREAD

Trace Types
No types assigned.

Traced Parameters

hstream=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 325 (0X0145)

Description
MTSH_ReturnFullEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
325 (0X0145)

Trace Groups
MTSHREAD

Trace Types
No types assigned.

Traced Parameters

hstream=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code:
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Description</th>
<th>Tracepoint</th>
<th>Minor Code</th>
<th>Description</th>
<th>Tracepoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>385 (0X0181)</td>
<td>MDM_mciSendStringExit</td>
<td>Static trace point in Multi-Media Extensions.</td>
<td>386 (0X0182)</td>
<td>MDM_mciSendCommandExit</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>387 (0X0183)</td>
<td>MDM_mciOpen_Exit</td>
<td>Static trace point in Multi-Media Extensions.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Traced Parameters:

- ID=%F RC=%F
- UserParm=%F
Multi-Media Extensions Major Code: 0X006D Minor Code: 387 (0X0183)

Static trace point in Multi-Media Extensions.

Minor Code
387 (0X0183)

Trace Groups
MCIOPEN

Trace Types
No types assigned.

Traced Parameters
ID=%F RC=%F UserParm=%F rc=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 388 (0X0184)

Description
MDM_MCDOpen_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
388 (0X0184)

Trace Groups
MCIOPEN

Trace Types
No types assigned.

Traced Parameters
ID=%F Type=%F RU=%F Class=%F UserParm=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 389 (0X0185)

Description
MDM_mciMakeActive_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
389 (0X0185)

Trace Groups
MCIOPEN

Trace Types
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 390 (0X0186)

Description
MDM_mciClose_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
390 (0X0186)

Trace Groups
MCICLOSE

Trace Types
No types assigned.

Traced Parameters
ID=%F Type=%F RU=%F Class=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 391 (0X0187)

Description
MDM_RESTORE_EXIT

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
391 (0X0187)

Trace Groups
MMRESMAN

Trace Types
No types assigned.

Traced Parameters
ID=%F Type=%F RU=%F UserParm=%F

--------------------------------------------
392 (0X0188)

**Description**  
MDM_SAVE_EXIT

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
392 (0X0188)

**Trace Groups**  
MMRESMAN

**Trace Types**  
No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 393 (0X0189)

**Description**  
MDM_Thread_call_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
393 (0X0189)

**Trace Groups**  
MMRESMAN

**Trace Types**  
No types assigned.

**Traced Parameters**

ID=%F Message=%F UserParm=%F RC=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 452 (0X01C4)

**Description**  
MTSH_GetEmptyExit

**Tracepoint**
### Multi-Media Extensions Major Code: 0X006D Minor Code: 452 (0X01C4)

- **Description**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 452 (0X01C4)
- **Trace Groups**: MTSHREAD
- **Trace Types**: No types assigned.
- **Traced Parameters**:
  - hstream=%F
  - RC=%F
  - ulFlags=%F

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 453 (0X01C5)

- **Description**: MTSH_ReturnFullExit
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 453 (0X01C5)
- **Trace Groups**: MTSHREAD
- **Trace Types**: No types assigned.
- **Traced Parameters**:
  - hstream=%F
  - RC=%F
  - ulFlags=%F

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 513 (0X0201)

- **Description**: MIO_DosOpen_Entry
- **Tracepoint**: Static trace point in Multi-Media Extensions.
- **Minor Code**: 513 (0X0201)
- **Trace Groups**: MMIO
- **Trace Types**: No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 514 (0X0202)

Description
MIO_DosRead_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
514 (0X0202)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 515 (0X0203)

Description
MIO_DosWrite_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
515 (0X0203)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code:
516 (0X0204)

**Description**
MIO_DosSeek_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
516 (0X0204)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

---------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 517 (0X0205)

**Description**
MIO_DosClose_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
517 (0X0205)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

---------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 518 (0X0206)

**Description**
MIO_DosDelete_Entry

**Tracepoint**
### Multi-Media Extensions Major Code: 0X006D Minor Code: 518 (0X0206)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemOpen_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>518 (0X0206)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 519 (0X0207)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemRead_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>519 (0X0207)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 520 (0X0208)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemRead_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>520 (0X0208)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
### Multi-Media Extensions Major Code: 0X006D Minor Code: 521 (0X0209)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>MIO_MemWrite_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>521 (0X0209)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 522 (0X020A)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>MIO_MemSeek_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>522 (0X020A)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code:
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Multi-Media Extensions Major Code: 0X006D Minor Code: 523 (0X020B)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>MIO_MemClose_Entry</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>523 (0X020B)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Multi-Media Extensions Major Code: 0X006D Minor Code: 524 (0X020C)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>MIO_mmioOpen_Entry</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>524 (0X020C)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Multi-Media Extensions Major Code: 0X006D Minor Code: 525 (0X020D)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>MIO_mmioClose_Entry</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
<td></td>
</tr>
</tbody>
</table>


Static trace point in Multi-Media Extensions.

**Minor Code**
525 (0X020D)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 526 (0X020E)

**Description**
MIO_mmioRead_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
526 (0X020E)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

cch=\%F

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 527 (0X020F)

**Description**
MIO_mmioWrite_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
527 (0X020F)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 528 (0X0210)

Description: MIO_mmioSeek_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 528 (0X0210)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>%F</td>
</tr>
<tr>
<td>Origin</td>
<td>%F</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 529 (0X0211)

Description: MIO_mmioFlush_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 529 (0X0211)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code:
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>Description</th>
<th>Tracepoint</th>
<th>Trace Groups</th>
<th>Trace Types</th>
<th>Traced Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>530 (0X0212)</td>
<td>MIO_mmioAscend_Entry</td>
<td>Static trace point in Multi-Media Extensions.</td>
<td>MMIO</td>
<td>No types assigned.</td>
<td></td>
</tr>
<tr>
<td>531 (0X0213)</td>
<td>MIO_mmioDescend_Entry</td>
<td>Static trace point in Multi-Media Extensions.</td>
<td>MMIO</td>
<td>No types assigned.</td>
<td></td>
</tr>
<tr>
<td>532 (0X0214)</td>
<td>MIO_mmioAdvance_Entry</td>
<td>Static trace point in Multi-Media Extensions.</td>
<td>MMIO</td>
<td>No types assigned.</td>
<td></td>
</tr>
</tbody>
</table>
Static trace point in Multi-Media Extensions.

**Minor Code**
532 (0X0214)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 533 (0X0215)

**Description**
MIO_mmioInstIOProc_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
533 (0X0215)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 534 (0X0216)

**Description**
MIO_mmioSendMsg_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
534 (0X0216)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 535 (0X0217)

Description: MIO_mmioAcquireSem_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 535 (0X0217)
Trace Groups: MMIO
Trace Types: No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 536 (0X0218)

Description: MIO_mmioDiscardSem_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 536 (0X0218)
Trace Groups: MMIO
Trace Types: No types assigned.
537 (0X0219)

Description: MIO_mmioCreateChunk_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 537 (0X0219)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

Multi-Media Extensions Major Code: 0X006D Minor Code: 538 (0X021A)

Description: MIO_mmioCFOpen_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 538 (0X021A)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

Multi-Media Extensions Major Code: 0X006D Minor Code: 539 (0X021B)

Description: MIO_mmioCFClose_Entry
Tracepoint:
Static trace point in Multi-Media Extensions.

**Minor Code**
539 (0X021B)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 540 (0X021C)

**Description**
MIO_mmioCFRmvShrEnt_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
540 (0X021C)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 541 (0X021D)

**Description**
MIO_mmioCFAddShrEnt_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
541 (0X021D)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 542 (0X021E)

Description
MIO_mmioCFSetInfo_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
542 (0X021E)

Trace Groups
MMIO

Trace Types
No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 543 (0X021F)

Description
MIO_mmioCFGetInfo_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
543 (0X021F)

Trace Groups
MMIO

Trace Types
No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code:
544 (0X0220)

Description: MIO_mmioCFOpnTmpElem_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 544 (0X0220)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

-------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 545 (0X0221)

Description: MIO_mmioCFClsTmpElem_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 545 (0X0221)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

-------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 546 (0X0222)

Description: MIO_mmioCFAddEnt_Entry
Tracepoint:
## Multi-Media Extensions

### Minor Code

546 (0X0222)

### Trace Groups

MMIO

### Trace Types

No types assigned.

### Traced Parameters

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 547 (0X0223)

#### Description

MIO_mmioCFChgEnt_Entry

#### Tracepoint

Static trace point in Multi-Media Extensions.

#### Minor Code

547 (0X0223)

#### Trace Groups

MMIO

#### Trace Types

No types assigned.

#### Traced Parameters

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 548 (0X0224)

#### Description

MIO_mmioCFCopy_Entry

#### Tracepoint

Static trace point in Multi-Media Extensions.

#### Minor Code

548 (0X0224)

#### Trace Groups

MMIO

#### Trace Types

No types assigned.
Multi-Media Extensions Major Code: 0X006D Minor Code: 549 (0X0225)

**Description**
MIO_mmioCFDelEnt_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
549 (0X0225)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 550 (0X0226)

**Description**
MIO_mmioCFAddElem_Entry

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
550 (0X0226)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

---

Multi-Media Extensions Major Code: 0X006D Minor Code:
551 (0X0227)

Description: MIO_mmioCFFndEnt_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 551 (0X0227)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 552 (0X0228)

Description: MIO_mmioIdentFile_Entry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 552 (0X0228)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 553 (0X0229)

Description: MIO_MidiOpen_Entry
Tracepoint:
### Multi-Media Extensions Major Code: 0X006D Minor Code: 553 (0X0229)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MidiRead_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 554 (0X022A)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MidiWrite_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 555 (0X022B)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MidiWrite_Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>553 (0X0229)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>554 (0X022A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>555 (0X022B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 556 (0X022C)

Description
MIO_MidiSeek_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
556 (0X022C)

Trace Groups
MMIO

Trace Types
No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 557 (0X022D)

Description
MIO_MidiClose_Entry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
557 (0X022D)

Trace Groups
MMIO

Trace Types
No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code:
576 (0X0240)

**Description**  
MSH_RdPlayList_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
576 (0X0240)

**Trace Groups**  
MSH

**Trace Types**  
No types assigned.

**Traced Parameters**

```
Operation=%F Operand1=%F Operand2=%F Operand3=%F Entry#=%F
```

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 577 (0X0241)

**Description**  
MSH_WrPlayList_Entry

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
577 (0X0241)

**Trace Groups**  
MSH

**Trace Types**  
No types assigned.

**Traced Parameters**

```
Operation=%F Operand1=%F Operand2=%F Operand3=%F Entry#=%F
```

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 592 (0X0250)

**Description**  
SWVR_GetImageEntry

**Tracepoint**
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>592 (0x0250)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

---

**Multi-Media Extensions Major Code: 0X006D Minor Code: 593 (0X0251)**

<table>
<thead>
<tr>
<th>Description</th>
<th>SWVR_CompIdxFrameEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>593 (0x0251)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

---

**Multi-Media Extensions Major Code: 0X006D Minor Code: 594 (0X0252)**

<table>
<thead>
<tr>
<th>Description</th>
<th>SWVR_CompRefrFrameEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>594 (0x0252)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
<tr>
<td>Multi-Media Extensions Major Code: 0X006D Minor Code: 595 (0X0253)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SWVR_PostBufferFullEntry</td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>595 (0X0253)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-Media Extensions Major Code: 0X006D Minor Code: 596 (0X0254)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tracepoint</strong></td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-Media Extensions Major Code: 0X006D Minor Code: 597 (0X0255)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Tracepoint</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Minor Code</td>
</tr>
<tr>
<td>Trace Groups</td>
</tr>
<tr>
<td>Trace Types</td>
</tr>
<tr>
<td>Traced Parameters</td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 600 (0X0258)

Description  
SWVR_BufferWriteStartEntry

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
600 (0X0258)

Trace Groups  
MMIO

Trace Types  
No types assigned.

Traced Parameters  

----------

Multi-Media Extensions Major Code: 0X006D Minor Code: 601 (0X0259)

Description  
SWVR_VCADosDevIOCTLEntry

Tracepoint  
Static trace point in Multi-Media Extensions.

Minor Code  
601 (0X0259)

Trace Groups  
MMIO

Trace Types  
No types assigned.

Traced Parameters  
No parameters traced.

----------

Multi-Media Extensions Major Code: 0X006D Minor Code: 602 (0X025A)
<table>
<thead>
<tr>
<th>Description</th>
<th>SWVR_VCABufferCopyEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>602 (0X025A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 603 (0X025B)

<table>
<thead>
<tr>
<th>Description</th>
<th>SWVR_VCAtoX4CopyEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>603 (0X025B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 604 (0X025C)

<table>
<thead>
<tr>
<th>Description</th>
<th>SWVR_VCAtoX2CopyEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>604 (0X025C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 605 (0X025D)

Description
SWVR_VCA640CopyEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
605 (0X025D)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 612 (0X0264)

Description
SVSH_DecompBufferEntry

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
612 (0X0264)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
hStream=%F Frame%=%F mmtSlave=%F mmtMaster=%F FramInterval=%F

Multi-Media Extensions Major Code: 0X006D Minor Code:
613 (0X0265)

Description: ULDC_DLLDroppedEntry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 613 (0X0265)
Trace Groups: SVSHSYNC
Trace Types: No types assigned.
Traced Parameters:
  Frame#=%F Tolerance=%F mmtNxtFrame=%F mmtCurrent=%F SyncMethod=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 617 (0X0269)

Description: SVMC_SetBitmapEntry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 617 (0X0269)
Trace Groups: SVSHSYNC
Trace Types: No types assigned.
Traced Parameters: No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 618 (0X026A)

Description: SVMC_BitBltEntry
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code 618 (0X026A)
Trace Groups SVSHSYNC
Trace Types No types assigned.
Traced Parameters No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 619 (0X026B)

Description SVMC_BlitUpEntry
Tracepoint Static trace point in Multi-Media Extensions.
Minor Code 619 (0X026B)
Trace Groups SVSHSYNC
Trace Types No types assigned.
Traced Parameters No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 620 (0X026C)

Description SVMC_SpiMoveCoorEntry
Tracepoint Static trace point in Multi-Media Extensions.
Minor Code 620 (0X026C)
Trace Groups SVSHSYNC
Trace Types No types assigned.
Traced Parameters No parameters traced.
Multi-Media Extensions Major Code: 0X006D Minor Code: 621 (0X026D)

Description

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 621 (0X026D)

Trace Groups: SVSHSYNC

Trace Types: No types assigned.

Traced Parameters: No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 641 (0X0281)

Description: MIO_DosOpen_Exit

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 641 (0X0281)

Trace Groups: MMIO

Trace Types: No types assigned.

Traced Parameters: 

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 642 (0X0282)

Description: MIO_DosRead_Exit

Tracepoint: 

Minor Code: 

Trace Groups: 

Trace Types: 

Traced Parameters: 

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
642 (0X0282)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 643 (0X0283)

Description
MIO_DosWrite_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
643 (0X0283)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 644 (0X0284)

Description
MIO_DosSeek_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
644 (0X0284)

Trace Groups
MMIO
Multi-Media Extensions Major Code: 0X006D Minor Code: 645 (0X0285)

Description: MIO_DosClose.Exit

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 645 (0X0285)

Trace Groups: MMIO

Trace Types: No types assigned.

Multi-Media Extensions Major Code: 0X006D Minor Code: 646 (0X0286)

Description: MIO_DosDelete.Exit

Tracepoint: Static trace point in Multi-Media Extensions.

Minor Code: 646 (0X0286)

Trace Groups: MMIO

Trace Types: No types assigned.
### Multi-Media Extensions Major Code: 0X006D Minor Code: 647 (0X0287)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemOpen.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>647 (0X0287)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 648 (0X0288)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemRead.Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>648 (0X0288)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 649 (0X0289)
<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemWrite_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>649 (0X0289)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemSeek_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>650 (0X028A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemClose_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>651 (0X028B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 650 (0X028A)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemSeek_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>650 (0X028A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>

Multi-Media Extensions Major Code: 0X006D Minor Code: 651 (0X028B)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MemClose_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>651 (0X028B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
</tbody>
</table>
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 652 (0X028C)

**Description**
MIO_mmioOpen_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
652 (0X028C)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 653 (0X028D)

**Description**
MIO_mmioClose_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
653 (0X028D)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
Multi-Media Extensions Major Code: 0X006D Minor Code: 654 (0X028E)

**Description**
MIO_mmioRead_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
654 (0X028E)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
rc=%F

------------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 655 (0X028F)

**Description**
MIO_mmioWrite_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
655 (0X028F)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
rc=%F

------------------------------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 656 (0X0290)
Multi-Media Extensions Major Code: 0X006D Minor Code: 656 (0X0290)

**Description**  
MIO_mmioSeek_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
656 (0X0290)

**Trace Groups**  
MMIO

**Trace Types**  
No types assigned.

**Traced Parameters**

$lFilePos=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 657 (0X0291)

**Description**  
MIO_mmioFlush_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
657 (0X0291)

**Trace Groups**  
MMIO

**Trace Types**  
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 658 (0X0292)

**Description**  
MIO_mmioAscend_Exit

**Tracepoint**  
Static trace point in Multi-Media Extensions.

**Minor Code**  
658 (0X0292)

**Trace Groups**
Multi-Media Extensions Major Code: 0X006D Minor Code: 659 (0X0293)

Description
MIO_mmioDescend.Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
659 (0X0293)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
Multi-Media Extensions Major Code: 0X006D Minor Code: 661 (0X0295)

Description: MIO_mmioInstIOProc_Exit
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 661 (0X0295)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

Multi-Media Extensions Major Code: 0X006D Minor Code: 662 (0X0296)

Description: MIO_mmioSendMsg_Exit
Tracepoint: Static trace point in Multi-Media Extensions.
Minor Code: 662 (0X0296)
Trace Groups: MMIO
Trace Types: No types assigned.
Traced Parameters:

Multi-Media Extensions Major Code: 0X006D Minor Code: 663 (0X0297)
Description | MIO_mmioAquireSem_Exit
---|---
Tracepoint | Static trace point in Multi-Media Extensions.
Minor Code | 663 (0X0297)
Trace Groups | MMIO
Trace Types | No types assigned.
Traced Parameters | 

Multi-Media Extensions Major Code: 0X006D Minor Code: 664 (0X0298)

Description | MIO_mmioDiscardSem_Exit
---|---
Tracepoint | Static trace point in Multi-Media Extensions.
Minor Code | 664 (0X0298)
Trace Groups | MMIO
Trace Types | No types assigned.
Traced Parameters | 

Multi-Media Extensions Major Code: 0X006D Minor Code: 665 (0X0299)

Description | MIO_mmioCreateChunk_Exit
---|---
Tracepoint | Static trace point in Multi-Media Extensions.
Minor Code | 665 (0X0299)
Trace Groups | 

---
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 666 (0X029A)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_mmioCFOpen_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>666 (0X029A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 667 (0X029B)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_mmioCFClose_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>667 (0X029B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 668 (0X029C)

**Description**
MIO_mmioCFRmvShrEnt_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
668 (0X029C)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 669 (0X029D)

**Description**
MIO_mmioCFAddShrEnt_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
669 (0X029D)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 670 (0X029E)
### MIO_mmioCFSetInfo_Exit

**Description:**
MIO_mmioCFSetInfo_Exit

**Tracepoint:**
Static trace point in Multi-Media Extensions.

**Minor Code:**
670 (0X029E)

**Trace Groups:**
MMIO

**Trace Types:**
No types assigned.

**Traced Parameters:**

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 671 (0X029F)

**Description:**
MIO_mmioCFGetInfo_Exit

**Tracepoint:**
Static trace point in Multi-Media Extensions.

**Minor Code:**
671 (0X029F)

**Trace Groups:**
MMIO

**Trace Types:**
No types assigned.

**Traced Parameters:**

---

### Multi-Media Extensions Major Code: 0X006D Minor Code: 672 (0X02A0)

**Description:**
MIO_mmioCFOpnTmpElem_Exit

**Tracepoint:**
Static trace point in Multi-Media Extensions.

**Minor Code:**
672 (0X02A0)

**Trace Groups:**

---
Multi-Media Extensions Major Code: 0X006D Minor Code: 673 (0X02A1)

Description
MIO_mmioCFClsTmpElem_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
673 (0X02A1)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 674 (0X02A2)

Description
MIO_mmioCFAddEnt_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
674 (0X02A2)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
Multi-Media Extensions Major Code: 0X006D Minor Code: 675 (0X02A3)

**Description**
MIO_mmioCFChgEnt_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
675 (0X02A3)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 676 (0X02A4)

**Description**
MIO_mmioCFCopy_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
676 (0X02A4)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 677 (0X02A5)
<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_mmioCFDelEnt_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>677 (0X02A5)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 678 (0X02A6)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_mmioCFAddElem_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>678 (0X02A6)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 679 (0X02A7)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_mmioCFFndEnt_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>679 (0X02A7)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 680 (0X02A8)

**Description**
MIO_mmioidentFile_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
680 (0X02A8)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**

Multi-Media Extensions Major Code: 0X006D Minor Code: 681 (0X02A9)

**Description**
MIO_MidiOpen_Exit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
681 (0X02A9)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
### Multi-Media Extensions Major Code: 0X006D Minor Code: 682 (0X02AA)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MidiRead_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>682 (0X02AA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 683 (0X02AB)

<table>
<thead>
<tr>
<th>Description</th>
<th>MIO_MidiWrite_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>683 (0X02AB)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>MMIO</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Media Extensions Major Code: 0X006D Minor Code: 684 (0X02AC)
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>MIO_MidiSeek_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>684 (0X02AC)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 685 (0X02AD)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>MIO_MidiClose_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>685 (0X02AD)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>MMIO</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

Multi-Media Extensions Major Code: 0X006D Minor Code: 704 (0X02C0)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>MSH_RdPlayList_Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>704 (0X02C0)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td></td>
</tr>
</tbody>
</table>
Multi-Media Extensions Major Code: 0X006D Minor Code: 705 (0X02C1)

Description
MSH_WrPlayList_Exit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
705 (0X02C1)

Trace Groups
MSH

Trace Types
No types assigned.

Traced Parameters
Operation=%F Operand1=%F Operand2=%F Operand3=%F rc=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 720 (0X02D0)

Description
SWVR_GetImageExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
720 (0X02D0)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.
Multi-Media Extensions Major Code: 0X006D Minor Code: 721 (0X02D1)

Description
SWVR_ComplIdxFrameExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
721 (0X02D1)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 722 (0X02D2)

Description
SWVR_CompRefrFrameExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
722 (0X02D2)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 723 (0X02D3)

Description
SWVR_PostBufferFullExit

Tracepoint
Static trace point in Multi-Media Extensions.

**Minor Code**
723 (0X02D3)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 724 (0X02D4)

**Description**
SWVR_PostBufferEmptyExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
724 (0X02D4)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 725 (0X02D5)

**Description**
SWVR_WaitMsgBufferFullExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
725 (0X02D5)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.
Multi-Media Extensions Major Code: 0X006D Minor Code: 726 (0X02D6)

Description
SWVR_WaitMsgBufferEmptyExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
726 (0X02D6)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 727 (0X02D7)

Description
SWVR_PostCountErrorExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
727 (0X02D7)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
PostCount=%d=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 728 (0X02D8)

Description
SWVR_BufferWriteDoneExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
728 (0X02D8)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
ByteCount=%d=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 729 (0X02D9)

**Description**
SWVR_VCADosDevIOCTLExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
729 (0X02D9)

**Trace Groups**
MMIO

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 730 (0X02DA)

**Description**
SWVR_VCABufferCopyExit

**Tracepoint**
Static trace point in Multi-Media Extensions.

**Minor Code**
730 (0X02DA)

**Trace Groups**
MMIO

**Trace Types**
Multi-Media Extensions Major Code: 0X006D Minor Code: 731 (0X02DB)

Description
SWVR_VCAtoX4CopyExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
731 (0X02DB)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 732 (0X02DC)

Description
SWVR_VCAtoX2CopyExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
732 (0X02DC)

Trace Groups
MMIO

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 733 (0X02DD)
**Description**  SWVR_VCA640CopyExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  733 (0X02DD)

**Trace Groups**  MMIO

**Trace Types**  No types assigned.

**Traced Parameters**  No parameters traced.

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 740 (0X02E4)

**Description**  SVSH_DecompBufferExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  740 (0X02E4)

**Trace Groups**  MMIO

**Trace Types**  No types assigned.

**Traced Parameters**

hStream=%F rc=%F BufLen=%F BlockIntvl=%F

--------------------------------------------

Multi-Media Extensions Major Code: 0X006D Minor Code: 741 (0X02E5)

**Description**  ULDC_DLLDroppedExit

**Tracepoint**  Static trace point in Multi-Media Extensions.

**Minor Code**  741 (0X02E5)
Multi-Media Extensions Major Code: 0X006D Minor Code: 745 (0X02E9)

Description
SVMC_SetBitmapExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
745 (0X02E9)

Trace Groups
SVSHSYNC

Trace Types
No types assigned.

Traced Parameters
No parameters traced.

Multi-Media Extensions Major Code: 0X006D Minor Code: 746 (0X02EA)

Description
SVMC_BitBltExit

Tracepoint
Static trace point in Multi-Media Extensions.

Minor Code
746 (0X02EA)

Trace Groups
SVSHSYNC

Trace Types
No types assigned.

Traced Parameters
No parameters traced.
### Multi-Media Extensions Major Code: 0X006D Minor Code: 747 (0X02EB)

<table>
<thead>
<tr>
<th>Description</th>
<th>SVMC_BlitUpExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>747 (0X02EB)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SVSHSYNC</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

--------------------------------------------

### Multi-Media Extensions Major Code: 0X006D Minor Code: 748 (0X02EC)

<table>
<thead>
<tr>
<th>Description</th>
<th>SVMC_SpiMoveCoorExit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Static trace point in Multi-Media Extensions.</td>
</tr>
<tr>
<td>Minor Code</td>
<td>748 (0X02EC)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SVSHSYNC</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>No parameters traced.</td>
</tr>
</tbody>
</table>

--------------------------------------------

### PMSHAPI.DLL Trace Events

The tracepoints for the PMSHAPI.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

#### Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

#### Trace events for PMSHAPI Major Code: 0X00C0, sorted by minor code.
Trace events for PMSHAPI Major Code: 0X00C0, sorted by tracepoint.

Trace Events for PMSHAPI Major Code: 0X00C0, Sorted by Minor Code

00001 (0X0001) ShePIInitialise Pre-Invocation
00002 (0X0002) WinQueryProfileInt Pre-Invocation
00003 (0X0003) WinQueryProfileString Pre-Invocation
00004 (0X0004) WinWriteProfileString Pre-Invocation
00005 (0X0005) WinQueryProfileSize Pre-Invocation
00006 (0X0006) WinQueryProfileData Pre-Invocation
00007 (0X0007) WinWriteProfileData Pre-Invocation
00008 (0X0008) WinIniSessionMgr Pre-Invocation
00009 (0X0009) Win16SetFgndWindow Pre-Invocation
00010 (0X000A) WinAddProgram Pre-Invocation
00011 (0X000B) WinRemoveProgram Pre-Invocation
00012 (0X000C) WinChangeProgram Pre-Invocation
00013 (0X000D) WinQueryDefinition Pre-Invocation
00014 (0X000E) WinQueryProgramTitles Pre-Invocation
00015 (0X000F) WinCreateGroup Pre-Invocation
00016 (0X0010) WinInitializePL Pre-Invocation
00017 (0X0011) WinQueryProgramUse Pre-Invocation
00018 (0X0012) WinQueryProgramType Pre-Invocation
00019 (0X0013) EntryProc Pre-Invocation
00020 (0X0014) PMExecRegister Pre-Invocation
00021 (0X0015) SheVioModeWait Pre-Invocation
00022 (0X0016) SheVioSavRedrawWait Pre-Invocation
00023 (0X0017) StartSaveWaitThreads Pre-Invocation
00024 (0X0018) CreateThreadStack Pre-Invocation
00025 (0X0019) RemoveOS2INI Pre-Invocation
00028 (0X001B) WinQueryProgramUse Pre-Invocation
00029 (0X001C) InitMinimizeIcon Pre-Invocation
00030 (0X001D) WSHInit Pre-Invocation
00031 (0X001E) WinAddProgram Pre-Invocation
00032 (0X001F) WinRemoveProgram Pre-Invocation
00033 (0X0020) SheInitializeIniFile Pre-Invocation
00034 (0X0021) InitialiseIniFile Pre-Invocation
00035 (0X0022) SetDosWarning Pre-Invocation
00036 (0X0023) CleanUp Pre-Invocation
00037 (0X0024) lpfnShellWndProc Pre-Invocation
32839 (0X8047) WinQuerySessionTitle Post-Invocation
32840 (0X8048) WinAddSwitchEntry Post-Invocation
32841 (0X8049) WinChangeSwitchEntry Post-Invocation
32854 (0X8056) WinQuerySwitchEntry Post-Invocation
32855 (0X8057) ShellIniItalizeIniFile Post-Invocation
32856 (0X8058) InitialiZeIniFile Post-Invocation
32857 (0X8059) SetDosWarning Post-Invocation
32864 (0X8060) CleanUp Post-Invocation
32865 (0X8061) ShellPostMessage Post-Invocation
32866 (0X8062) ShellSendMessage Post-Invocation
32867 (0X8063) strtm Post-Invocation
32868 (0X8064) BadSwitch Post-Invocation
32869 (0X8065) WinQueryTaskTitle Post-Invocation
32870 (0X8066) WinQueryTaskSizePost Post-Invocation
32871 (0X8067) WinQuerySwitchList Post-Invocation
32872 (0X8068) WinRemoveSwitchEntry Post-Invocation
32873 (0X8069) WinSwitchToProgram Post-Invocation
32874 (0X806A) WinSwitchProgramRegister Post-Invocation
32875 (0X806B) FindSwitchEntry Post-Invocation
32876 (0X806C) WinEndProgram Post-Invocation
32877 (0X806D) WinStopProgram Post-Invocation
32878 (0X806E) WinEndWindowSession Post-Invocation
32879 (0X806F) lpfnShellWndProc Post-Invocation
32880 (0X8070) lpfnIconWndProc Post-Invocation
32881 (0X8071) WinSwitchToTaskManager Post-Invocation
32882 (0X8072) WinSetTitle Post-Invocation
32883 (0X8073) SetKBDHotKey Post-Invocation
32884 (0X8074) WinPMFILERegister Post-Invocation
32885 (0X8075) WinInitSession Post-Invocation
32886 (0X8076) WinEndSession Post-Invocation
32887 (0X8077) WinInitSwEntry Post-Invocation
32888 (0X8078) WinSetSwEntry Post-Invocation
32889 (0X8079) WinSetExtIdFocus Post-Invocation
32890 (0X807A) WinQueryExtIdFocus Post-Invocation
32891 (0X807B) WinQueryExtIdFocus Post-Invocation
32892 (0X807C) WinQueryExtIdFocus Post-Invocation
32893 (0X807D) WinQueryExtIdFocus Post-Invocation
32894 (0X807E) WinQueryExtIdFocus Post-Invocation
32895 (0X807F) WinQueryExtIdFocus Post-Invocation
32896 (0X8080) WinQueryExtIdFocus Post-Invocation
32897 (0X8081) WinQueryExtIdFocus Post-Invocation
32898 (0X8082) WinQueryExtIdFocus Post-Invocation
32899 (0X8083) WinQueryExtIdFocus Post-Invocation
32900 (0X8084) WinQueryExtIdFocus Post-Invocation
32901 (0X8085) WinQueryExtIdFocus Post-Invocation
32902 (0X8086) WinQueryExtIdFocus Post-Invocation
32903 (0X8087) WinQueryExtIdFocus Post-Invocation
32904 (0X8088) WinQueryExtIdFocus Post-Invocation
32905 (0X8089) WinQueryExtIdFocus Post-Invocation
32906 (0X808A) WinQueryExtIdFocus Post-Invocation
32907 (0X808B) WinQueryExtIdFocus Post-Invocation
32908 (0X808C) WinQueryExtIdFocus Post-Invocation
32909 (0X808D) WinQueryExtIdFocus Post-Invocation
32910 (0X808E) WinQueryExtIdFocus Post-Invocation
32911 (0X808F) WinQueryExtIdFocus Post-Invocation
32912 (0X8090) WinQueryExtIdFocus Post-Invocation
32913 (0X8091) WinQueryExtIdFocus Post-Invocation
32914 (0X8092) WinQueryExtIdFocus Post-Invocation
32915 (0X8093) WinQueryExtIdFocus Post-Invocation
32916 (0X8094) WinQueryExtIdFocus Post-Invocation
32917 (0X8095) WinQueryExtIdFocus Post-Invocation
32918 (0X8096) WinQueryExtIdFocus Post-Invocation
32919 (0X8097) WinQueryExtIdFocus Post-Invocation
32920 (0X8098) WinQueryExtIdFocus Post-Invocation

--------------------------------------------------------------------------

Trace Events for PMSHAPI Major Code: 0X00C0, Sorted by Tracepoint

ACTIVATESESSION 00067 (0X0043)
ActivateSession 32835 (0X8043)
BADSWITCH 00100 (0X0064)
BadSwitch 32868 (0X8064)
CLEANUP 00096 (0X0060)
CleanUp 32864 (0X8060)
.CREATETHREADSTACK 00055 (0X0037)
CreateThreadStack 32864 (0X8060)
ENTRYPROC 00041 (0X0029)
EnterProc 32807 (0X8027)
EXITPROC 00041 (0X0029)
ExitProc 32809 (0X8029)
FINDSWITCHENTRY 00113 (0X0071)
FindSwitchEntry 32881 (0X8071)
FNBADAPPDLGPROC 00121 (0X0079)
FindSwitchEntry 32881 (0X8071)
INITIALISEFILE 00088 (0X0058)
InitializeSessionManager 00150 (0X0080)
INITMINIMIZEICON 00048 (0X0030)

--------------------------------------------------------------------------
WinQueryExtIdFocus 32902 (0X8086)
WinSetExtIdFocus 32903 (0X8087)
WinSetSwEntry 32901 (0X8085)
WinSetTitle 32915 (0X8093)
WinSwitchToProgram2 32896 (0X8080)
WinSwitchToTaskManager 32888 (0X8078)
fnBadAppDlgProc 32889 (0X8079)
lpfnIconWndProc 32887 (0X8077)
lpfnShellWndProc 32886 (0X8076)
strtrn 32867 (0X8063)

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 1 (0X0001)

Description  
ShePIInitialise Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.SHE16PIINITIALISE

Minor Code  
1 (0X0001)

Trace Groups  
SHAPI

Trace Types  
PRE

Traced Parameters

   pPILocation = %a
   pPILocation -> %s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 2 (0X0002)

Description  
WinQueryProfileInt Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILEINT

Minor Code  
2 (0X0002)

Trace Groups  
SHAPI

Trace Types  
PRE

Traced Parameters

   Default = %w, pKeyName = %a
   pAppName = %a, hab = %a
PMHSAPI Major Code: 0X00C0 Minor Code: 3 (0X0003)

Description
WinQueryProfileString Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMHSAPI.WIN16QUERYPROFILESTRING

Minor Code
3 (0X0003)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

BuffSize = %w, pBuffer = %a
pDefault = %a, pKeyName = %a
pAppName = %a, hab = %a
pDefault -> %s
pKeyName -> %s
pAppName -> %s

PMHSAPI Major Code: 0X00C0 Minor Code: 4 (0X0004)

Description
WinWriteProfileString Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMHSAPI.WIN16WRITEPROFILESTRING

Minor Code
4 (0X0004)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

pASCIIz = %a, pKeyName = %a
PMSHAPI Major Code: 0X00C0 Minor Code: 5 (0X0005)

Description
WinQueryProfileSize Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILESIZE

Minor Code
5 (0X0005)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

pSize = %a, pKeyName = %a
pAppName = %a, hab = %a
pKeyName -> %s
pAppName -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 6 (0X0006)

Description
WinQueryProfileData Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILEDATA

Minor Code
6 (0X0006)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

pBufferSize = %a, pBuffer = %a, pKeyName = %a
PMISHAPI Major Code: 0X00C0 Minor Code: 7 (0X0007)

Description
WinWriteProfileData Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16WRITEPROFILEDATA

Minor Code
7 (0X0007)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
Length = %w, pData = %a, pKeyName = %a
pAppName = %a, hab = %a
pKeyName -> %s
pAppName -> %s

PMISHAPI Major Code: 0X00C0 Minor Code: 8 (0X0008)

Description
WinInitSessionMgr Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INITSESSIONMGR

Minor Code
8 (0X0008)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
Return Address = %w
PMSHAPI Major Code: 0X00C0 Minor Code: 9 (0X0009)

Description: Win16SetFgndWindow Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SETFGNDWINDOW
Minor Code: 9 (0X0009)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:

sgid = %w, hwndfocus=%d

PMSHAPI Major Code: 0X00C0 Minor Code: 18 (0X0012)

Description: WinAddProgram Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ADDPROGRAM
Minor Code: 18 (0X0012)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:

GrpHandle = %a, BufPtr = %a, hab = %a
progt -> %w, Title/Icon/Exec = %u

PMSHAPI Major Code: 0X00C0 Minor Code: 19 (0X0013)

Description: WinRemoveProgram Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16REMOVEPROGRAM

**Minor Code**
19 (0X0013)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

PrgHandle = %a

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 20 (0X0014)

**Description**
WinChangeProgram Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CHANGEPROGRAM

**Minor Code**
20 (0X0014)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

BufPtr = %a, TrgHandle = %a
progt -> %w, Title/Icon/Exec = %u

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 21 (0X0015)

**Description**
WinQueryDefinition Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYDEFINITION

**Minor Code**
21 (0X0015)

**Trace Groups**
SHAPI

**Trace Types**
PRE
Traced Parameters

InLength = %w, pibble = %a
TargetHandle = %a, hab = %a

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 22 (0X0016)

Description
ExecuteStartUp Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.EXECUTESTARTUP

Minor Code
22 (0X0016)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

%Return Address = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 22 (0X0016)

Description
WinQueryProgramTitles Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMTITLES

Minor Code
22 (0X0016)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

pTotal = %a, cbBuffer = %w, pBuffer = %a
TargetHandle = %a, hab = %a

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 23 (0X0017)
WinCreateGroup Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CREATEGROUP

Minor Code
23 (0X0017)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Rhip = %d, Rth = %d, vis = %w, InString = %a
hab = %a
InString -> %s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 24 (0X0018)

WinInitializePL Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INITIALIZEPL

Minor Code
24 (0X0018)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

fOptions=%F

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32 (0X0020)

WinQueryProgramUse Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMUSE

Minor Code
**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

- Exe = %a
- Exe -> %s

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 35 (0X0023)**

**Description**

WinDestroyGroup Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16DESTROYGROUP

**Minor Code**

35 (0X0023)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

GrpHandle = %a

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 37 (0X0025)**

**Description**

ShlLoadFont Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SHLLOADFONT

**Minor Code**

37 (0X0025)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

pFontName -> %s
PMSHAPI Major Code: 0X00C0 Minor Code: 38 (0X0026)

**Description**
WinQueryProgramType Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMTYPE

**Minor Code**
38 (0x0026)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

Exe = %a
Exe -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 39 (0X0027)

**Description**
EntryProc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.ENTRYPROC

**Minor Code**
39 (0x0027)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

fForeground = %w, hswitch = %d

PMSHAPI Major Code: 0X00C0 Minor Code: 40 (0X0028)

**Description**
PMExecRegister Pre-Invocation
Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.PM16EXCREGISTER

Minor Code  
40 (0X0028)

Trace Groups  
SHAPI

Trace Types  
PRE

Traced Parameters

hwnd = %a

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 41 (0X0029)

Description  
ExitProc Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.EXITPROC

Minor Code  
41 (0X0029)

Trace Groups  
SHAPI

Trace Types  
PRE

Traced Parameters

mp2 = %d, mp1 = %d, message = %w, hwnd = %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 48 (0X0030)

Description  
InitMinimizeIcon Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.INITMINIMIZEICON

Minor Code  
48 (0X0030)

Trace Groups  
SHAPI

Trace Types  
PRE

Traced Parameters
pswctl = %w, pszIconFile = %a
pszIconFile -> %s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 51 (0X0033)

Description
WSHInit Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WSHINIT

Minor Code
51 (0X0033)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

dStack_AR = %a

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 52 (0X0034)

Description
SheVioModeWait Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.SHEVIOMODEWAIT

Minor Code
52 (0X0034)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Return Address = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 53 (0X0035)
SheVioSavRedrawWait Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.SHEVIOSAVREDRAWWAIT

Minor Code
53 (0X0035)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Return Address = %w

-----------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 54 (0X0036)

StartSaveWaitThreads Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.STARTSAVEWAITTHREADS

Minor Code
54 (0X0036)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Return Address = %w

-----------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 55 (0X0037)

CreateThreadStack Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.CREATETHREADSTACK

Minor Code
55 (0X0037)

Trace Groups
SHAPI

Trace Types
PRE
Traced Parameters

StackSize = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 56 (0X0038)

Description
RemoveOS2INI Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.REMOVEOS2INI

Minor Code
56 (0X0038)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Return Address = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 65 (0X0041)

Description
ReportNoHardErrors Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.REPORTNOHARDERRORS

Minor Code
65 (0X0041)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

pszTitle = %a, hwnd = %d
pszTitle -> %s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 66 (0X0042)
<table>
<thead>
<tr>
<th>Description</th>
<th>SwitchToNextSession Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.SWITCHTONEXTSESSION</td>
</tr>
<tr>
<td>Minor Code</td>
<td>66 (0X0042)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>fPrev = %w, fsIn = %w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 67 (0X0043)

<table>
<thead>
<tr>
<th>Description</th>
<th>ActivateSession Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.ACTIVATESESSION</td>
</tr>
<tr>
<td>Minor Code</td>
<td>67 (0X0043)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>fs = %w, hwndToBottom = %d, pswi = %w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 68 (0X0044)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinInstStartApp Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INSTSTARTAPP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>68 (0X0044)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
</tbody>
</table>


Trace Types
PRE

Traced Parameters
fsOptions = %w, pData = %a, pszCmdLine = %a, aszApplication = %a
cCount = %w, hwndNotifyWindow = %d, hini = %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 69 (0X0045)

Description
WinTerminateApp Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16TERMINATEAPP

Minor Code
69 (0X0045)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
happ = %d %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 70 (0X0046)

Description
WinCreateSwitchEntry Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CREATESWITCHENTRY

Minor Code
70 (0X0046)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
lpswctl = %a, habIgnored = %w

--------------------------------------------
PMSHAPI Major Code: 0X00C0 Minor Code: 71 (0X0047)

**Description**
WinQuerySessionTitle Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSESSIONTITLE

**Minor Code**
71 (0X0047)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

- cbNameBufferLength = %d,
- szNameBuffer = %a, sid = %w,
- szNameBuffer -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 72 (0X0048)

**Description**
WinAddSwitchEntry Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ADDSWITCHENTRY

**Minor Code**
72 (0X0048)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

- lpswctl = %a

PMSHAPI Major Code: 0X00C0 Minor Code: 73 (0X0049)

**Description**
WinChangeSwitchEntry Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CHANGESWITCHENTRY

**Minor Code**
73 (0X0049)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

```
lpswctl = %a, hswitch = %d
```

PMSHAPI Major Code: 0X00C0 Minor Code: 86 (0X0056)

**Description**
WinQuerySwitchEntry Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSWITCHENTRY

**Minor Code**
86 (0X0056)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

```
lpswctl = %a, hswitch = %d
```

PMSHAPI Major Code: 0X00C0 Minor Code: 87 (0X0057)

**Description**
ShellInitializeIniFile Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.SHE16INITIALIZEINIFILE

**Minor Code**
87 (0X0057)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

```
lpswctl = %a, hswitch = %d
```
PMSHAPI Major Code: 0X00C0 Minor Code: 88 (0X0058)

**Description**
InitialiseIniFile Pre-Invocatio

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.INITIALISEINIFILE

**Minor Code**
88 (0X0058)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

Return Address = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 89 (0X0059)

**Description**
SetDosWarning Pre-Invocatio

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.SETDOSWARNING

**Minor Code**
89 (0X0059)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

Rc = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 96 (0X0060)

**Description**
CleanUp Pre-Invocatio
Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.CLEANUP

Minor Code
96 (0X0060)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
Rc = %w

-------------------------------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 97 (0X0061)

Description
ShellPostMessage Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.SHELLPOSTMESSAGE

Minor Code
97 (0X0061)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
pswi = %w, mp1 = %d

-------------------------------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 98 (0X0062)

Description
ShellSendMessage Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.SHELLSENDMESSAGE

Minor Code
98 (0X0062)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
PMSHAPI Major Code: 0X00C0 Minor Code: 99 (0X0063)

Description
strrn Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.STRTRN

Minor Code
99 (0X0063)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

cch = %w, pszTo = %a, pszFrom = %a
pszTo -> %s
pszFrom -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 100 (0X0064)

Description
BadSwitch Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.BADSWITCH

Minor Code
100 (0X0064)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

hswitch = %d

PMSHAPI Major Code: 0X00C0 Minor Code: 101 (0X0065)
WinQueryTaskTitle Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYTASKTITLE

Minor Code
101 (0X0065)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

\texttt{\textbackslash cbNameBufferLength = \%w,}
\texttt{\textbackslash szNameBuffer = \%a, sid = \%w}
\texttt{\textbackslash szNameBuffer -> \%s}

PMSHAPI Major Code: 0X00C0 Minor Code: 102 (0X0066)

WinQueryTaskSizePos Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYTASKSIZEPOS

Minor Code
102 (0X0066)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

\texttt{\textbackslash pswp = \%a, sid = \%w, habIgnored = \%w}

PMSHAPI Major Code: 0X00C0 Minor Code: 103 (0X0067)

WinQuerySwitchList Pre-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSWITCHLIST

Minor Code
103 (0X0067)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
    cbBufferLength = %w, pswBlock = %a,

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 104 (0X0068)

Description: WinRemoveSwitchEntry Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WIN16REMOVESWITCHENTRY
Minor Code: 104 (0X0068)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
    hswitch = %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 105 (0X0069)

Description: WinSwitchToProgram Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SWITCHTOPROGRAM
Minor Code: 105 (0X0069)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
    hswitch = %d

--------------------------------------------
PMSHAPI Major Code: 0X00C0 Minor Code: 112 (0X0070)

**Description**  
WinSwitchProgramRegister Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SWITCHPROGRAMREGISTER

**Minor Code**  
112 (0X0070)

**Trace Groups**  
SHAPI

**Trace Types**  
PRE

**Traced Parameters**

\[
pfnwp = %a, hmq = %d, hwnd = %d
\]


PMSHAPI Major Code: 0X00C0 Minor Code: 113 (0X0071)

**Description**  
FindSwitchEntry Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.FINDSWITCHENTRY

**Minor Code**  
113 (0X0071)

**Trace Groups**  
SHAPI

**Trace Types**  
PRE

**Traced Parameters**

\[
fs = %w, sid = %w
\]


PMSHAPI Major Code: 0X00C0 Minor Code: 115 (0X0073)

**Description**  
WinEndProgram Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ENDPROGRAM

**Minor Code**
115 (0X0073)

**Trace Groups**
- SHAPI

**Trace Types**
- PRE

**Traced Parameters**
- idProcess = %w, hswitch = %d

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 116 (0X0074)**

**Description**
- WinStopProgram Pre-Invocation

**Tracepoint**
- Public symbol defined dynamic tracepoint: PMSHAPI.WIN16STOPPROGRAM

**Minor Code**
- 116 (0X0074)

**Trace Groups**
- SHAPI

**Trace Types**
- PRE

**Traced Parameters**
- idProcess = %w, hswitch = %d

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 117 (0X0075)**

**Description**
- WinEndWindowSession Pre-Invocation

**Tracepoint**
- Public symbol defined dynamic tracepoint: PMSHAPI.WINENDWINDOWSESSION

**Minor Code**
- 117 (0X0075)

**Trace Groups**
- SHAPI

**Trace Types**
- PRE

**Traced Parameters**
- hwnd = %d
## PMSHAPI Major Code: 0X00C0 Minor Code: 118 (0X0076)

<table>
<thead>
<tr>
<th>Description</th>
<th>lpfnShellWndProc Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.LPFNHELLWNDPROC</td>
</tr>
<tr>
<td>Minor Code</td>
<td>118 (0X0076)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>mp2 = %d, mp1 = %d, message = %w, hwnd = %d</td>
</tr>
</tbody>
</table>

## PMSHAPI Major Code: 0X00C0 Minor Code: 119 (0X0077)

<table>
<thead>
<tr>
<th>Description</th>
<th>lpfnIconWndProc Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.LPFNICONWNDPROC</td>
</tr>
<tr>
<td>Minor Code</td>
<td>119 (0X0077)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>mp2 = %d, mp1 = %d, message = %w, hwnd = %d</td>
</tr>
</tbody>
</table>

## PMSHAPI Major Code: 0X00C0 Minor Code: 120 (0X0078)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinSwitchToTaskManager Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WINSWITCHTOTASKMANAGER</td>
</tr>
</tbody>
</table>
PMSHAPI Major Code: 0X00C0 Minor Code: 120 (0X0078)

Description: fnBadAppDlgProc Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.FNBADAPDPOLGPROC

Traced Parameters:
- Return Address = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 121 (0X0079)

Description: fnBadAppDlgProc Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.FNBADAPDPOLGPROC

Minor Code: 121 (0X0079)

Trace Groups: SHAPI

Trace Types: PRE

Traced Parameters:
- mp2 = %d, mp1 = %d, message = %w, hwnd = %d

PMSHAPI Major Code: 0X00C0 Minor Code: 128 (0X0080)

Description: WinSwitchToProgram2 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WINSWITCHTOPROGRAM2

Minor Code: 128 (0X0080)

Trace Groups: SHAPI

Trace Types: PRE

Traced Parameters:
- fs = %w, hswitch = %d
PMSHAPI Major Code: 0x00C0 Minor Code: 129 (0x0081)

**Description**
WinProcessHotKey Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WINPROCESSHOTKEY

**Minor Code**
129 (0x0081)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**
fProcess = %w, pqmsg = %a

PMSHAPI Major Code: 0x00C0 Minor Code: 130 (0x0082)

**Description**
WinInitSession Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WININITSESSION

**Minor Code**
130 (0x0082)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**
pReqBlock = %a, ppfn = %a

PMSHAPI Major Code: 0x00C0 Minor Code: 131 (0x0083)

**Description**
WinEndSession Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WINENDSESSION
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>0x0083 (131)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>sidEnded = %w</td>
<td></td>
</tr>
</tbody>
</table>

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 0x0084**

<table>
<thead>
<tr>
<th>Description</th>
<th>WinInitSwEntry Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WININITSWENTRY</td>
</tr>
<tr>
<td>Minor Code</td>
<td>0x0084 (132)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>Return Address = %w</td>
<td></td>
</tr>
</tbody>
</table>

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 0x0085**

<table>
<thead>
<tr>
<th>Description</th>
<th>WinSetSwEntry Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WINSETSWENTRY</td>
</tr>
<tr>
<td>Minor Code</td>
<td>0x0085 (133)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>pReqBlock = %a, ppfn = %a</td>
<td></td>
</tr>
</tbody>
</table>
PMSHAPI Major Code: 0X00C0 Minor Code: 134 (0X0086)

**Description**
WinQueryExtIdFocus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WINQUERYEXTIDFOCUS

**Minor Code**
134 (0X0086)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

pSessionId = %F

PMSHAPI Major Code: 0X00C0 Minor Code: 135 (0X0087)

**Description**
WinSetExtIdFocus Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WINSETTEXTIDFOCUS

**Minor Code**
135 (0X0087)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

hSwitch = %d
sid = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 136 (0X0088)

**Description**
SheSystemShutdown Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.SHE16SYSTEMSHUTDOWN

Minor Code: 136 (0X0088)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
Return Address = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 137 (0X0089)

Description: Start16SystemExecutables Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.START16SYSTEMEXECUTABLES

Minor Code: 137 (0X0089)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
%Return Address = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 144 (0X0090)

Description: ShlLoadPublicFonts Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.SHL16LOADPUBLICFONTS

Minor Code: 144 (0X0090)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:
pszFontDLL -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 145 (0X0091)

Description
WinNoShutdown Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16NOSHUTDOWN
Minor Code
145 (0X0091)
Trace Groups
SHAPI
Trace Types
PRE
Traced Parameters
flag = %w, sid = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 147 (0X0093)

Description
WinSetTitle Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WINSETTITLE
Minor Code
147 (0X0093)
Trace Groups
SHAPI
Trace Types
PRE
Traced Parameters
pNewTitle = %a
pNewTitle -> %s

PMSHAPI Major Code: 0X00C0 Minor Code: 148 (0X0094)

Description
WinCPLRegister Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CPLREGISTER

**Minor Code**
148 (0X0094)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

hwnd = %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 149 (0X0095)

**Description**
WinPMFILERegister Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16PMFILEREGISTER

**Minor Code**
149 (0X0095)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

hwnd = %d

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 150 (0X0096)

**Description**
InitialiseSessionManager Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.INITIALISESESSIONMANAGER

**Minor Code**
150 (0X0096)

**Trace Groups**
SHAPI

**Trace Types**
PRE
Traced Parameters

Return Address = %w

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 151 (0X0097)

Description
SetKBDHotKey Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.SETKBDHOTKEY

Minor Code
151 (0X0097)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

Return Address = %w

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 257 (0X0101)

Description
PrfQueryProfileSize Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILESIZE

Minor Code
257 (0X0101)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

hini=%F, pszApp=%F, pszKey=%F, pulReqLen=%F
App=%s, Key=%s

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 258 (0X0102)
<table>
<thead>
<tr>
<th>Description</th>
<th>PrfOpenProfile Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.PRF32OPENPROFILE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>258 (0X0102)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hab =%F, pszFileName=%f, FileName=%s</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>PrfCloseProfile Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CLOSEPROFILE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>259 (0X0103)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hini = %F</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>PrfRemoveProgram Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.PRF32REMOVEPROGRAM</td>
</tr>
<tr>
<td>Minor Code</td>
<td>260 (0X0104)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
</tbody>
</table>
Trace Types
PRE

Traced Parameters
hini=%F, ProgHandle=%F

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 261 (0X0105)

Description
PrfCreateGroup Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CREATEGROUP

Minor Code
261 (0X0105)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
hini=%F, pszTitle=%F, chVisibility=%b, Title=%s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 262 (0X0106)

Description
PrfDestroyGroup Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32DESTROYGROUP

Minor Code
262 (0X0106)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters
hini=%F, hprogGroup=%F

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 263 (0X0107)
Description  PrfQueryProfile Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILE
Minor Code  263 (0X0107)
Trace Groups  SHAPI
Trace Types  PRE
Traced Parameters

hab=%F, pPrfProfile=%

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 264 (0X0108)

Description  PrfReset Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMSHAPI.PRF32RESET
Minor Code  264 (0X0108)
Trace Groups  SHAPI
Trace Types  PRE
Traced Parameters

hab=%, pPrfProfile=%

cchUserName=%F, pszUserName=%F

cchSysName =%F, pszSysName=%F

UserName=%s, SysName=%s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 265 (0X0109)

Description  PrfAddProgram Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMSHAPI.PRF32ADDPROGRAM
**Minor Code**

265 (0X0109)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

\[
\begin{align*}
\text{hini} & = \%F, \text{pDetails} = \%F, \text{hprogGroup} = \%F \\
\text{pDetails.length} & = \%F, \text{progt.category} = \%F, \text{progt.fbVisible} = \%F \\
\text{pszTitle} & = \%F, \text{pszExecutable} = \%F, \text{pszParameters} = \%F \\
\text{pszStartupDir} & = \%F, \text{pszIcon} = \%F, \text{pszEnvironment} = \%F \\
\text{swpInit.fl} & = \%F, \text{cy} = \%F, \text{cx} = \%F \\
\text{y} & = \%F, \text{x} = \%F, \text{hwndInsertB} = \%F \\
\text{hwnd} & = \%F, \text{ulRes1} = \%F, \text{ulRes2} = \%F \\
\text{Title} & = \%s, \text{Executable} = \%s, \text{Parameters} = \%s \\
\text{StartupDir} & = \%s, \text{Icon} = \%s, \text{Environment} = \%s
\end{align*}
\]

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 272 (0X0110)**

**Description**

PrfChangeProgram Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CHANGEPROGRAM

**Minor Code**

272 (0X0110)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

\[
\begin{align*}
\text{hini} & = \%F, \text{pDetails} = \%F, \text{hprogGroup} = \%F \\
\text{pDetails.length} & = \%F, \text{progt.category} = \%F, \text{progt.fbVisible} = \%F \\
\text{pszTitle} & = \%F, \text{pszExecutable} = \%F, \text{pszParameters} = \%F \\
\text{pszStartupDir} & = \%F, \text{pszIcon} = \%F, \text{pszEnvironment} = \%F \\
\text{swpInit.fl} & = \%F, \text{cy} = \%F, \text{cx} = \%F \\
\text{y} & = \%F, \text{x} = \%F, \text{hwndInsertB} = \%F \\
\text{hwnd} & = \%F, \text{ulRes1} = \%F, \text{ulRes2} = \%F
\end{align*}
\]
PMSHAPI Major Code: 0X00C0 Minor Code: 273 (0X0111)

**Description**
PrfQueryDefinition Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYDEFINITION

**Minor Code**
273 (0X0111)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

hini=%F, hprog=%F, pDetails=%F, cchBufferMax=%F

PMSHAPI Major Code: 0X00C0 Minor Code: 274 (0X0112)

**Description**
PrfQueryProgramHandle Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMHANDLE

**Minor Code**
274 (0X0112)

**Trace Groups**
SHAPI

**Trace Types**
PRE

**Traced Parameters**

hini=%F, pszExeName=%F, phpga=%F, cb=%F
pcHandles=%F, ExeName=%s

PMSHAPI Major Code: 0X00C0 Minor Code: 275 (0X0113)
**Description**: PrfQueryProgramTitles Pre-Invocation

**Tracepoint**: Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMTITLES

**Minor Code**: 275 (0X0113)

**Trace Groups**: SHAPI

**Trace Types**: PRE

**Traced Parameters**:

- hini = %F, hprogGroup = %F, pTitles = %F
- cchBufferMax = %F, pulCount = %F

--------------------------------------------

**PMSHAPI Major Code**: 0X00C0 **Minor Code**: 277 (0X0115)

**Description**: PrfQueryProfileString Pre-Invocation

**Tracepoint**: Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILESTRING

**Minor Code**: 277 (0X0115)

**Trace Groups**: SHAPI

**Trace Types**: PRE

**Traced Parameters**:

- hini = %F, pszApp = %f, pszKey = %f
- pszDefault = %f, pBuffer = %f, cchBufferMax = %F
- App = %s, Key = %s, Default = %s

--------------------------------------------

**PMSHAPI Major Code**: 0X00C0 **Minor Code**: 278 (0X0116)

**Description**: PrfWriteProfileString Pre-Invocation

**Tracepoint**: Public symbol defined dynamic tracepoint: PMSHAPI.PRF32WRITEPROFILESTRING

**Minor Code**: 278 (0X0116)
Trace Groups: SHAPI

Trace Types: PRE

Traced Parameters:

hini=%F, pszApp=%f, pszKey=%f, pszData=%F
App=%s
Key=%s
Data=%s

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 280 (0X0118)

Description: PrfWriteProfileData Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.PRF32WRITEPROFILEDATA
Minor Code: 280 (0X0118)
Trace Groups: SHAPI
Trace Types: PRE
Traced Parameters:

hini=%F, pszApp=%f, pszKey=%f, pBuf=%F, cchDataLen=%F
App=%s, Key=%s
Data=%u

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 306 (0X0132)

Description: PrfQueryProgramCategory Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMCATEGORY
Minor Code: 306 (0X0132)
Trace Groups: SHAPI
Trace Types
PRE

Traced Parameters

hini=%F, pszExe=%F, Exe=%s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 4375 (0X1117)

Description
PrfQueryProfileData Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILEDATA

Minor Code
4375 (0X1117)

Trace Groups
SHAPI

Trace Types
PRE

Traced Parameters

hini=%F, pszApp=%F, pszKey=%F, pBuf=%F, pulBuflen=%F
Buflen=%F, App=%s, Key=%s

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32769 (0X8001)

Description
ShePIInitialise Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.She16PIInitialise

Minor Code
32769 (0X8001)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32770 (0X8002)
WinQueryProfileInt Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileInt

Minor Code 32770 (0X8002)

Trace Groups SHAPI

Trace Types POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32771 (0X8003)

WinSetFgndWindow Post-invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileString

Minor Code 32771 (0X8003)

Trace Groups SHAPI

Trace Types POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32772 (0X8004)

WinWriteProfileString Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16WriteProfileString

Minor Code 32772 (0X8004)

Trace Groups SHAPI
Trace Types: POST

Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32773 (0X8005)

Description: WinSetForegroundWindow Post-invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.W16QueryProfileSize
Minor Code: 32773 (0X8005)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32774 (0X8006)

Description: WinQueryProfileData Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.W16QueryProfileData
Minor Code: 32774 (0X8006)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32775 (0X8007)
<table>
<thead>
<tr>
<th>Description</th>
<th>WinWriteProfileData Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16WriteProfileData</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32775 (0X8007)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32776 (0X8008)**

<table>
<thead>
<tr>
<th>Description</th>
<th>WinInitSessionMgr Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16InitSessionMgr</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32776 (0X8008)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32777 (0X8009)**

<table>
<thead>
<tr>
<th>Description</th>
<th>WinSetFgndWindow Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16SetFgndWindow</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32777 (0X8009)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
</tbody>
</table>
Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32786 (0X8012)

Description

WinAddProgram Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.Win16AddProgram

Minor Code

32786 (0X8012)

Trace Groups

SHAPI

Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32787 (0X8013)

Description

WinRemoveProgram Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.Win16RemoveProgram

Minor Code

32787 (0X8013)

Trace Groups

SHAPI

Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32788 (0X8014)
<table>
<thead>
<tr>
<th>Description</th>
<th>WinChangeProgram Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16ChangeProgram</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32788 (0X8014)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32789 (0X8015)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinQueryDefinition Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryDefinition</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32789 (0X8015)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32790 (0X8016)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinQueryProgramTitles Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramTitles</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32790 (0X8016)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>SHAPI</td>
</tr>
</tbody>
</table>
Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32791 (0X8017)

Description: WinCreateGroup Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.Win16CreateGroup

Minor Code: 32791 (0X8017)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32792 (0X8018)

Description: WinInitializePL Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.Win16InitializePL

Minor Code: 32792 (0X8018)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32800 (0X8020)
WinQueryProgramUse Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramUse

Minor Code
32800 (0X8020)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32803 (0X8023)

WinDestroyGroup Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16DestroyGroup

Minor Code
32803 (0X8023)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32805 (0X8025)

ShlLoadFont Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.ShlLoadFont

Minor Code
32805 (0X8025)

Trace Groups
SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32806 (0X8026)

Description: WinQueryProgramType Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramType
Minor Code: 32806 (0X8026)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32807 (0X8027)

Description: EntryProc Post-invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.EntryProc
Minor Code: 32807 (0X8027)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32809 (0X8029)
<table>
<thead>
<tr>
<th>Description</th>
<th>ExitProc Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.ExitProc</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32809 (0X8029)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32816 (0X8030)

<table>
<thead>
<tr>
<th>Description</th>
<th>InitMinimizeIcon Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.InitMinimizeIcon</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32816 (0X8030)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

---------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32819 (0X8033)

<table>
<thead>
<tr>
<th>Description</th>
<th>WSHInit Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WSHInit</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32819 (0X8033)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
</tbody>
</table>
**Trace Types**  
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32820 (0X8034)**

**Description**  
SheVioModeWait Post-invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.SheVioModeWait

**Minor Code**  
32820 (0X8034)

**Trace Groups**  
SHAPI

**Trace Types**  
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32821 (0X8035)**

**Description**  
SheVioSavRedrawWait Post-invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.SheVioSavRedrawWait

**Minor Code**  
32821 (0X8035)

**Trace Groups**  
SHAPI

**Trace Types**  
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32822 (0X8036)**
**PMSHAPI Major Code: 0X00C0 Minor Code: 32822 (0X8036)**

**Description**
StartSaveWaitThreads Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.StartSaveWaitThreads

**Minor Code**
32822 (0X8036)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32823 (0X8037)**

**Description**
CreateThreadStack Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.CreateThreadStack

**Minor Code**
32823 (0X8037)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32824 (0X8038)**

**Description**
RemoveOS2INI Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.RemoveOS2INI

**Minor Code**
32824 (0X8038)

**Trace Groups**
SHAPI
Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32833 (0X8041)

Description
ReportNoHardErrors Post-invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.ReportNoHardErrors

Minor Code
32833 (0X8041)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32834 (0X8042)

Description
SwitchToNextSession Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.SwitchToNextSession

Minor Code
32834 (0X8042)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32835 (0X8043)
ActivateSession Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.ActivateSession

Minor Code: 32835 (0X8043)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:

Return Code = %w

-----------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32836 (0X8044)

Description: WinInstStartApp Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16InstStartApp

Minor Code: 32836 (0X8044)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:

Return Code = %w

-----------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32837 (0X8045)

Description: WinTerminateApp Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16TerminateApp

Minor Code: 32837 (0X8045)
Trace Groups: SHAPI
**Trace Types**

POST

**Trace Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32838 (0X8046)**

**Description**

WinCreateSwitchEntry Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.Win16CreateSwitchEntry

**Minor Code**

32838 (0X8046)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32839 (0X8047)**

**Description**

WinQuerySessionTitle Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySessionTitle

**Minor Code**

32839 (0X8047)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32840 (0X8048)**
WinAddSwitchEntry Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16AddSwitchEntry

Minor Code: 32840 (0X8048)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32841 (0X8049)

WinChangeSwitchEntry Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16ChangeSwitchEntry

Minor Code: 32841 (0X8049)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32854 (0X8056)

WinQuerySwitchEntry Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySwitchEntry

Minor Code: 32854 (0X8056)

Trace Groups: SHAPI
Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32855 (0X8057)

Description
SheInitializeIniFile Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.She16InitializeIniFile

Minor Code
32855 (0X8057)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32856 (0X8058)

Description
InitialiseIniFile Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.InitialiseIniFile

Minor Code
32856 (0X8058)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32857 (0X8059)
PMSHAPI Major Code: 0X00C0 Minor Code: 32857 (0X8059)

Description: SetDosWarning Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.SetDosWarning
Minor Code: 32857 (0X8059)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32864 (0X8060)

Description: CleanUp Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.CleanUp
Minor Code: 32864 (0X8060)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32865 (0X8061)

Description: ShellPostMessage Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.ShellPostMessage
Minor Code: 32865 (0X8061)
Trace Groups: SHAPI
Trace Types: POST

Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32866 (0X8062)

Description: ShellSendMessage Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.ShellSendMessage
Minor Code: 32866 (0X8062)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32867 (0X8063)

Description: strtrn Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.strtrn
Minor Code: 32867 (0X8063)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:
Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32868 (0X8064)
**BadSwitch Post-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.BadSwitch

**Minor Code**
32868 (0X8064)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**WinQueryTaskTitle Post-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskTitle

**Minor Code**
32869 (0X8065)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**WinQueryTaskSizePos Post-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskSizePos

**Minor Code**
32870 (0X8066)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32869 (0X8065)**

**Description**
WinQueryTaskTitle Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskTitle

**Minor Code**
32869 (0X8065)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32870 (0X8066)**

**Description**
WinQueryTaskSizePos Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskSizePos

**Minor Code**
32870 (0X8066)

**Trace Groups**
SHAPI
Trace Types
  POST

Traced Parameters
  Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32871 (0X8067)

Description
  WinQuerySwitchList Post-Invocation

Tracepoint
  Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySwitchList

Minor Code
  32871 (0X8067)

Trace Groups
  SHAPI

Trace Types
  POST

Traced Parameters
  Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32872 (0X8068)

Description
  WinRemoveSwitchEntry Post-Invocation

Tracepoint
  Public symbol defined dynamic tracepoint: PMSHAPI.Win16RemoveSwitchEntry

Minor Code
  32872 (0X8068)

Trace Groups
  SHAPI

Trace Types
  POST

Traced Parameters
  Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32873 (0X8069)
**Description**
WinSwitchToProgram Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16SwitchToProgram

**Minor Code**
32873 (0X8069)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32880 (0X8070)**

**Description**
WinSwitchProgramRegister Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.Win16SwitchProgramRegister

**Minor Code**
32880 (0X8070)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**

Return Code = %w

--------------------------------------------

**PMSHAPI Major Code: 0X00C0 Minor Code: 32881 (0X8071)**

**Description**
FindSwitchEntry Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.FindSwitchEntry

**Minor Code**
32881 (0X8071)

**Trace Groups**
SHAPI
Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32883 (0X8073)

Description

WinEndProgram Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.Win16EndProgram

Minor Code

32883 (0X8073)

Trace Groups

SHAPI

Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32884 (0X8074)

Description

WinStopProgram Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.Win16StopProgram

Minor Code

32884 (0X8074)

Trace Groups

SHAPI

Trace Types

POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32885 (0X8075)
WinEndWindowSession Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.WinEndWindowSession

Minor Code: 32885 (0X8075)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:

Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32886 (0X8076)

lpfnShellWndProc Post-invocation

Public symbol defined dynamic tracepoint: PMSHAPI.lpfnShellWndProc

Minor Code: 32886 (0X8076)
Trace Groups: SHAPI
Trace Types: POST
Traced Parameters:

Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32887 (0X8077)

lpfnIconWndProc Post-invocation

Public symbol defined dynamic tracepoint: PMSHAPI.lpfnIconWndProc

Minor Code: 32887 (0X8077)
Trace Groups: SHAPI
Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32888 (0X8078)

Description
WinSwitchToTaskManager Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WinSwitchToTaskManager

Minor Code
32888 (0X8078)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32889 (0X8079)

Description
fnBadAppDlgProc Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.fnBadAppDlgProc

Minor Code
32889 (0X8079)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32896 (0X8080)
WinSwitchToProgram2 Post-Invocation

Description
Public symbol defined dynamic tracepoint: PMSHAPI.WinSwitchToProgram2

Minor Code
32896 (0X8080)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32898 (0X8082)

Description
WinInitSession Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WinInitSession

Minor Code
32898 (0X8082)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32899 (0X8083)

Description
WinEndSession Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.WinEndSession

Minor Code
32899 (0X8083)

Trace Groups
SHAPI
Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32900 (0X8084)

Description: WinInitSwEntry Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WinInitSwEntry

Minor Code: 32900 (0X8084)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32901 (0X8085)

Description: WinSetSwEntry Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSHAPI.WinSetSwEntry

Minor Code: 32901 (0X8085)

Trace Groups: SHAPI

Trace Types: POST

Traced Parameters:

Return Code = %w

--------------------------------------------

PMSHAPI Major Code: 0X00C0 Minor Code: 32902 (0X8086)
**Description**
WinQueryExtIdFocus Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WinQueryExtIdFocus

**Minor Code**
32902 (0X8086)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**Description**
WinSetExtIdFocus Post-invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.WinSetExtIdFocus

**Minor Code**
32903 (0X8087)

**Trace Groups**
SHAPI

**Trace Types**
POST

**Traced Parameters**
Return Code = %w

--------------------------------------------

**Description**
SheSystemShutdown Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSHAPI.She16SystemShutdown

**Minor Code**
32904 (0X8088)

**Trace Groups**
SHAPI
Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32905 (0X8089)

Description
StartSystemExecutables Post-invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.Start16SystemExecutables

Minor Code
32905 (0X8089)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32912 (0X8090)

Description
Shl16LoadPublicFonts Post-invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSHAPI.Shl16LoadPublicFonts

Minor Code
32912 (0X8090)

Trace Groups
SHAPI

Trace Types
POST

Traced Parameters
Return Code = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 32913 (0X8091)
<table>
<thead>
<tr>
<th>Description</th>
<th>WinNoShutdown Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16NoShutdown</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32913 (0X8091)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32915 (0X8093)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinSetTitle Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.WinSetTitle</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32915 (0X8093)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
<tr>
<td>Return Code</td>
<td>%w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32916 (0X8094)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinCPLRegister Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16CPLRegister</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32916 (0X8094)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Return Code = %w</td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32917 (0X8095)

<table>
<thead>
<tr>
<th>Description</th>
<th>WinPMFILERegister Post-invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.Win16PMFILERegister</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32917 (0X8095)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>POST</td>
</tr>
<tr>
<td>Return Code = %w</td>
<td></td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32918 (0X8096)

<table>
<thead>
<tr>
<th>Description</th>
<th>InitialiseSessionManager Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSHAPI.InitialiseSessionManager</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32918 (0X8096)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Trace Types</td>
<td>SHAPI</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>POST</td>
</tr>
<tr>
<td>Return Code = %w</td>
<td></td>
</tr>
</tbody>
</table>

PMSHAPI Major Code: 0X00C0 Minor Code: 32919 (0X8097)
SetKBDHotKey Post-Invocation

Public symbol defined dynamic tracepoint: PMSHAPI.SetKBDHotKey

32919 (0X8097)

SHAPI

POST

Return Code = %w

PMWIN.DLL Trace Events

The tracepoints for the PMWIN.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace Events for PMWIN Major Code: 0X00C2, Sorted by Minor Code

00100 (0X0064) WINREGISTERCLASS PRE-INVOCATION
00101 (0X0065) WINDEFWINDOWPROC PRE-INVOCATION
00102 (0X0066) WINDESTROYWINDOW PRE-INVOCATION
00103 (0X0067) WINSHOWWINDOW PRE-INVOCATION
00104 (0X0068) WINQUERYWINDOWRECT PRE-INVOCATION
00105 (0X0069) WINCREATEWINDOW PRE-INVOCATION
00106 (0X006A) WINCREATETEMPLATE WINCREATEWINDOW PRE-INVOCATION
00107 (0X006B) WINENABLEWINDOW PRE-INVOCATION
00108 (0X006C) WINISWINDOWENABLED PRE-INVOCATION
00109 (0X006D) WINISWINDOW PRE-INVOCATION
00110 (0X006E) WINISCHILD PRE-INVOCATION
00111 (0X006F) WINISCHILDWINDOW PRE-INVOCATION
00112 (0X0070) WINQUERYWINDOWTEXT PRE-INVOCATION
00113 (0X0071) WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION
00114 (0X0072) WINMULTIWINDOWNEXT FROMID PRE-INVOCATION
00115 (0X0073) WINMULTIWINDOWNEXT FROMID PRE-INVOCATION
00116 (0X0074) WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION
00117 (0X0075) WINISCHILD PRE-INVOCATION
00118 (0X0076) WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION
00119 (0X0077) WINQUERYOBJECTWINDOWTEXT LENGTH PRE-INVOCATION
00708 (0X02C4) WINBROADCASTMSG PRE-INVOCATION
00709 (0X02C5) WINWAITMSG PRE-INVOCATION
00710 (0X02C6) WINQUERYQUEUESTATUS PRE-INVOCATION
00711 (0X02C7) WINPOSTQUEUEMSG PRE-INVOCATION
00712 (0X02C8) WINQUERYMSGPOS PRE-INVOCATION
00713 (0X02C9) WINQUERYMSGTIME PRE-INVOCATION
00714 (0X02CA) WINMSGSEMWAIT PRE-INVOCATION
00715 (0X02CB) WINMSGMUXSEMWAIT PRE-INVOCATION
00717 (0X02CD) WINSETMSGINTEREST PRE-INVOCATION
00718 (0X02CE) WINSENDMSG PRE-INVOCATION
00751 (0X02EF) WINSENDOQUEUEMSG PRE-INVOCATION
00752 (0X02F0) CREATEQUEUE PRE-INVOCATION
00753 (0X02F1) FREEQUEUE PRE-INVOCATION
00754 (0X02F2) CLEARQUEUEGLOBALS PRE-INVOCATION
00755 (0X02F3) REASSIGNINPUT PRE-INVOCATION
00756 (0X02F4) ASSOCIATEQUEUE PRE-INVOCATION
00757 (0X02F5) READMESSAGE PRE-INVOCATION
00800 (0X0320) WINQUERYQUEUEINFO POST-INVOCATION
00801 (0X0321) WINCREATEMSGQUEUE POST-INVOCATION
00802 (0X0322) WINDESTROYMSGQUEUE POST-INVOCATION
00803 (0X0323) WINCANCELSHUTDOWN POST-INVOCATION
00804 (0X0324) WINGETMSG/WINPEEKMSG POST-INVOCATION
00805 (0X0325) WINPEEKMSG POST-INVOCATION
00806 (0X0326) WINDISPATCHMSG POST-INVOCATION
00807 (0X0327) WINPOSTMSG/WINPOSTQUEUEMSG POST-INVOCATION
00808 (0X0328) WINSENSENDMSG POST-INVOCATION
00809 (0X0329) WINBROADCASTMSG POST-INVOCATION
00810 (0X032A) WINWAITMSG POST-INVOCATION
00811 (0X032B) WINQUERYQUEUESTATUS POST-INVOCATION
00812 (0X032C) WINPOSTQUEUEMSG POST-INVOCATION
00813 (0X032D) WINQUERYMSGPOS POST-INVOCATION
00814 (0X032E) WINQUERYMSGTIME POST-INVOCATION
00815 (0X032F) WINMSGSEMWAIT POST-INVOCATION
00816 (0X0330) WINMSGSEMWAIT/WINMSGMUXSEMWAIT POST-INVOCATION
00817 (0X0331) WINSETMSGINTEREST POST-INVOCATION
00818 (0X0332) WINSENDMSG/WINSENDOQUEUEMSG POST-INVOCATION
00900 (0X0384) WINSETFOCUS PRE-INVOCATION
00901 (0X0385) WINFOCUSCHANGE PRE-INVOCATION
00902 (0X0386) WINSETCAPTURE PRE-INVOCATION
00903 (0X0387) WINFOCUCAPTURE PRE-INVOCATION
00904 (0X0388) WINGETFOCUS PRE-INVOCATION
00905 (0X0389) WINFOCUSWINDOW PRE-INVOCATION
00906 (0X038A) WINFOCUSACTIVETEMPLATE PRE-INVOCATION
00907 (0X038B) WINGETKEYSTATE PRE-INVOCATION
00908 (0X038C) WINGETPHYSKEYSTATE PRE-INVOCATION
00909 (0X038D) WINENABLEPHYSINPUT PRE-INVOCATION
00910 (0X038E) WINPHYSINPUTENABLED PRE-INVOCATION
00911 (0X038F) WINFOCUSTEMPLATE PRE-INVOCATION
00912 (0X0390) WINTRACKRECT PRE-INVOCATION
00913 (0X0391) WINSHOWTRACKRECT PRE-INVOCATION
01000 (0X03E8) WINSETFOCUSPOST INVOCATION
01001 (0X03E9) WINFOCUSCHANGEPRE INVOCATION
01002 (0X03EA) WINSETCAPTURE POST INVOCATION
01003 (0X03EB) WINQUERYFOCUS POST INVOCATION
01004 (0X03EC) WINQUERYFOCUSWINQUERYSYSMODALWINDOW POST INVOCATION
01005 (0X03ED) WINGETKEYSTATE POST INVOCATION
01006 (0X03EE) WINGETPHYSKEYSTATE POST INVOCATION
01007 (0X03EF) WINENABLEPHYSINPUT POST INVOCATION
01008 (0X03F0) WINPHYSINPUTENABLED POST INVOCATION
01009 (0X03F1) WINFOCUSTEMPLATE POST INVOCATION
01100 (0X044C) WINLOADDLL POST INVOCATION
01101 (0X044D) WINDLGBOX POST INVOCATION
01102 (0X044E) WINDISMISSDIALOG POST INVOCATION
01103 (0X044F) WINSETDGITEMSHORT POST INVOCATION
01104 (0X0450) WINQUERYDGITEMSHORT POST INVOCATION
01105 (0X0451) WINSETDGITEMTEXT POST INVOCATION
01106 (0X0452) WINQUERYDGITEMTEXT POST INVOCATION
01107 (0X0453) WINDEFDIALOGPROC PRE INVOCATION
01108 (0X0454) WINALARM POST INVOCATION
01109 (0X0455) WINMESSAGEBOX POST INVOCATION
01110 (0X0456) WINPROCESSDIALOG POST INVOCATION
01111 (0X0457) WINSENDSSLIGHTMESSAGE POST INVOCATION
01112 (0X0458) WINMAPDOLGPOINTS POST INVOCATION
02210 (0X08A2) WINQUERYCLIPBRDVIEWER POST-INVOCATION
02300 (0X08FC) WINDESTROYCURSOR PRE-INVOCATION
02301 (0X08FD) WINSHOWCURSOR PRE-INVOCATION
02302 (0X08FE) WINCREATECURSOR PRE-INVOCATION
02303 (0X08FF) WINQUERYCURSORINFO PRE-INVOCATION
02304 (0X0900) WINSETPOINTER PRE-INVOCATION
02305 (0X0901) WINSHOWPOINTER PRE-INVOCATION
02306 (0X0902) WINQUERYSYSPOINTER PRE-INVOCATION
02307 (0X0903) WINLOADPOINTER PRE-INVOCATION
02308 (0X0904) WINCREATEPOINTER PRE-INVOCATION
02309 (0X0905) WINDESTROYPOINTER PRE-INVOCATION
02310 (0X0906) WINQUERYPOINTER PRE-INVOCATION
02311 (0X0907) WINSETPOINTERPOS PRE-INVOCATION
02312 (0X0908) WINQUERYPOINTERPOS PRE-INVOCATION
02313 (0X0909) WINQUERYPOINTERINFO PRE-INVOCATION
02314 (0X090A) WINDRAWPOINTER PRE-INVOCATION
02315 (0X090B) WINGETSYSBITMAP PRE-INVOCATION
02400 (0X0960) WINDESTROYCURSOR POST-INVOCATION
02401 (0X0961) WINSHOWCURSOR POST-INVOCATION
02402 (0X0962) WINCREATECURSOR POST-INVOCATION
02403 (0X0963) WINQUERYCURSORINFO POST-INVOCATION
02404 (0X0964) WINSETPOINTER POST-INVOCATION
02405 (0X0965) WINSHOWPOINTER POST-INVOCATION
02406 (0X0966) WINQUERYSYSPOINTER POST-INVOCATION
02407 (0X0967) WINLOADPOINTER POST-INVOCATION
02408 (0X0968) WINCREATEPOINTER POST-INVOCATION
02409 (0X0969) WINDESTROYPOINTER POST-INVOCATION
02410 (0X096A) WINQUERYPOINTER POST-INVOCATION
02411 (0X096B) WINSETPOINTERPOS POST-INVOCATION
02412 (0X096C) WINQUERYPOINTERPOS POST-INVOCATION
02413 (0X096D) WINQUERYPOINTERINFO POST-INVOCATION
02414 (0X096E) WINDRAWPOINTER POST-INVOCATION
02415 (0X096F) WINGETSYSBITMAP POST-INVOCATION
02500 (0X09C4) WINSETHOOK PRE-INVOCATION
02501 (0X09C5) WINRELEASEHOOK PRE-INVOCATION
02502 (0X09C6) WNCALLMSGFILTER PRE-INVOCATION
02552 (0X09F8) FARCALLHOOK PRE-INVOCATION
02553 (0X09F9) FREEQUEUEWINDOWHOOKS PRE-INVOCATION
02600 (0X0A28) WINSETHOOK POST-INVOCATION
02601 (0X0A29) WINRELEASEHOOK POST-INVOCATION
02602 (0X0A2A) WNCALLMSGFILTER POST-INVOCATION
02700 (0X0ABC) WINSETCP PRE-INVOCATION
02701 (0X0ABD) WINQUERYCPC PRE-INVOCATION
02702 (0X0ABE) WINQUERYCLIPSTC PRE-INVOCATION
02703 (0X0ABF) WINCPTRANSALTESTRING PRE-INVOCATION
02704 (0X0AC0) WINCPTRANSALTECHAR PRE-INVOCATION
02705 (0X0AC1) WINUPPERPRE-INVOCATION
02706 (0X0AC2) WINUPPERCHAR PRE-INVOCATION
02707 (0X0AC3) WINNEXTCHAR PRE-INVOCATION
02708 (0X0AC4) WINPREVCHAR PRE-INVOCATION
02709 (0X0AC5) WINCOMPARESTRING PRE-INVOCATION
02710 (0X0AC6) WINLOADSTRING PRE-INVOCATION
02711 (0X0AC7) WINLOADMESSAGE PRE-INVOCATION
02750 (0X0ABE) WINLOADCHARXLATEETBL PRE-INVOCATION
02751 (0X0ABF) WINSETCHARXLATEETBL PRE-INVOCATION
02752 (0X0AC0) WINQUERYCHARXLATEETBL PRE-INVOCATION
02753 (0X0AC1) WINLOADVKEYGPHXLATEETBL PRE-INVOCATION
02754 (0X0AC2) WINSETVKEYGPHXLATEETBL PRE-INVOCATION
02755 (0X0AC3) WINQUERYVKEYGPHXLATEETBL PRE-INVOCATION
02756 (0X0AC4) WINSETKBDLAYOUT PRE-INVOCATION
02757 (0X0AC5) WINQUERYKBDLAYOUT PRE-INVOCATION
02800 (0X0AF0) WINSETCP POST-INVOCATION
02801 (0X0AF1) WINQUERYCPC POST-INVOCATION
02802 (0X0AF2) WINQUERYCLIPSTC POST-INVOCATION
02803 (0X0AF3) WINCPTRANSALTESTRING POST-INVOCATION
02804 (0X0AF4) WINCPTRANSALTECHAR POST-INVOCATION
02805 (0X0AF5) WINUPPER POST-INVOCATION
02806 (0X0AF6) WINUPPERCHAR POST-INVOCATION
02807 (0X0AF7) WINNEXTCHAR POST-INVOCATION
02808 (0X0AF8) WINPREVCHAR POST-INVOCATION
02809 (0X0AF9) WINCOMPARESTRING POST-INVOCATION
02900 (0X0B54) WINCREATEHEAP PRE-INVOCATION
02901 (0X0B55) WINDESTROYHEAP PRE-INVOCATION
Trace Events for PMWIN Major Code: 0X00C2, Sorted by Tracepoint

ASSOCIATEQUEUE 00756 (0X02F4)
CLEARQUEUEGLOBALS 00754 (0X02F2)
COMPACTMOVEABLEHEAP 02950 (0X0B86)
COPYWINDOWRECT 00162 (0X00A2)
CREATEQUEUE 00752 (0X02F0)
DELETELISTITEM 01358 (0X054E)
DESTROYLIST 01356 (0X054C)
DRAWTEXT 00408 (0X0198)
FARCALLHOOK 02552 (0X09F8)
FINDFREEBLOCK 02951 (0X0B87)
FINDMAXFREEBLOCK 02952 (0X0B88)
FINDTOPWINDOW 00163 (0X00A3)
FREEQUEUE 00753 (0X02F1)
FREEQUEUETWINDOWHOOKS 02553 (0X09F9)
GETLISTITEM 01360 (0X0550)
GETLISTITEMLENGTH 01359 (0X054F)
GETSIZEDS 02955 (0X0B8B)
INSERTFREEBLOCK 02953 (0X0B89)
INSERTLISTITEM 01357 (0X054D)
POSTWINALARM 01208 (0X04B8)
POSTWINBEGINENUMWINDOWS 00650 (0X028A)
POSTWINBEGINPAINT 00205 (0X00CD)
POSTWINBROADCASTMSG 00809 (0X0329)
POSTWINCALLHOOK 01804 (0X070C)
POSTWINCOMPARESTRINGS 02809 (0X0A9F)
POSTWINFOCOPYACCETABLE 02003 (0X07D3)
POSTWINCOPYRECT 01602 (0X0642)
POSTWINCREATEACCETABLE 02001 (0X07D1)
POSTWINCREATECURSOR 02402 (0X0962)
POSTWINCREATEFRAMECONTROLS 01403 (0X057B)
POSTWINCREATENOTIFIER 02408 (0X0968)
POSTWINDESTROYACCETABLE 02002 (0X07D2)
POSTWINDESTROYCURSOR 02400 (0X0960)
POSTWINDESTROYPOINTER 02409 (0X0969)
POSTWINDESTROYWINDOW 00202 (0X00CA)
POSTWINGETDISPATCHGROUP 00806 (0X0326)
POSTWINGETPOINTER 02414 (0X096E)
POSTWINGETPHYSINPUT 01007 (0X03EF)
POSTWINGETWINDOW 00215 (0X00D7)
POSTWINGETWINDOWDATE 00217 (0X00D9)
POSTWINGETNUMWINDOWS 00652 (0X028C)
POSTWINGETPAINT 00206 (0X00CE)
POSTWINNUMDLGITEM 01214 (0X04BE)
POSTWINQUALRECT 01603 (0X0643)
POSTWINEXCLUDEUPDATEREGION 00417 (0X01A1)
POSTWINFILLRECT 00212 (0X00D4)
POSTWINFOCUSCHANGE 01001 (0X03E9)
POSTWINGETCLIPPS 00208 (0X00D0)
WINMAPDLPPOINTS 01112 (0X0458)
WINMAPWINDOWPOINTS 00504 (0X01F8)
WINMESSAGEGEO 01109 (0X0455)
WINMSGMUXSEMWAIT 00715 (0X02CB)
WINMSGSEMWAIT 00714 (0X02CB)
WINMULTWINDOWFROMMIDS 00114 (0X0072)
WINNEXTCHAR 02707 (0X0A93)
WINOFFSETRECT 01505 (0X05E1)
WINOPENCLIPBRD 02100 (0X0834)
WINOPENWINDOWDC 00324 (0X0144)
WINPEEKMSG 00704 (0X02C0)
WINPOSTMSG 00706 (0X02C2)
WINPOSTQUEUEMSGS 00711 (0X02C7)
WINPREVCHAR 02708 (0X0A94)
WINPROCESSDLG 01110 (0X0456)
WINPTINRECT 01507 (0X05E3)
WINQUERYACCELTABLE 01906 (0X0772)
WINQUERYACTIVEWINDOW 00124 (0X007C)
WINQUERYCAPTURE 00903 (0X0387)
WINQUERYCHARXLATETBL 02752 (0X0AC0)
WINQUERYCLASSINFO 00123 (0X007B)
WINQUERYCLASSNAME 00122 (0X007A)
WINQUERYCLIPBRDDATA 02106 (0X083A)
WINQUERYCLIPBRDFMTINFO 02108 (0X083C)
WINQUERYCLIPBRDOWNER 02104 (0X0838)
WINQUERYCLIPBRDVIEWER 02110 (0X083E)
WINQUERYTCP 02701 (0X0A8D)
WINQUERYCP 02702 (0X0A8E)
WINQUERYCURSORINFO 02303 (0X0A8F)
WINQUERYDESKTOPWINDOW 00120 (0X007B)
WINQUERYDLGITEMSHORT 01104 (0X0450)
WINQUERYDLGITEMTEXT 01106 (0X0452)
WINQUERYFOCUS 00904 (0X0388)
WINQUERYKBDLAYOUT 02757 (0X0AC5)
WINQUERYMSGPOS 00712 (0X02CB)
WINQUERYMSGTIME 00713 (0X02CD)
WINQUERYOBJECTWINDOW 00119 (0X0077)
WINQUERYPOINTER 02310 (0X0906)
WINQUERYPOINTERINFO 02313 (0X0909)
WINQUERYPOINTERPOS 02312 (0X0908)
WINQUERYQUEUEINFO 00700 (0X02BC)
WINQUERYQUEUESTATUS 00710 (0X02C6)
WINQUERYSYSCOLOR 01706 (0X06AA)
WINQUERYSYSMODALWINDOW 00125 (0X007D)
WINQUERYSYSPINTER 02306 (0X0902)
WINQUERYSYSTEMATOMTABLE 01703 (0X06A7)
WINQUERYSYSVALUE 01704 (0X06A8)
WINQUERYUPDATERECT 00314 (0X013A)
WINQUERYUPDATEREGION 00315 (0X013B)
WINQUERYVERSION 01700 (0X06AA)
WINQUERYVKEYGLYPHXLATETBL 02755 (0X0AC3)
WINQUERYWINDOW 00116 (0X0074)
WINQUERYWINDOWDC 00312 (0X0138)
WINQUERYWINDOWLOCKCOUNT 00129 (0X0081)
WINQUERYWINDOWPOS 00301 (0X012D)
WINQUERYWINDOWPROCESS 00118 (0X0076)
WINQUERYWINDOWRECT 00104 (0X006B)
WINQUERYWINDOWTEXT 00110 (0X006E)
WINQUERYWINDOWTEXTLENGTH 00112 (0X0070)
WINQUERYWINDOWWULONG 00132 (0X0084)
WINQUERYWINDOWUSHORT 00130 (0X0082)
WINQUEUEDNDPROC 00153 (0X0099)
WINREALLOCMEM 02904 (0X0B58)
WINREGISTERCLASS 00100 (0X006D)
WINREGISTERWINDOWDESTROY 00128 (0X0080)
WINRELEASEHOOK 02501 (0X0905)
WINRELEASEPS 00323 (0X0143)
WINSAVEWINDOWPOS 00157 (0X009D)
WINSCROLLBARWNDPROC 01351 (0X0547)
WINSCROLLWINDOW 00325 (0X0145)
WINSENDDLGITEMMSGS 01111 (0X0457)
WINSENDMSG 00718 (0X02C2)
WINSENDQUEUEMSGS 00751 (0X02EF)
PMWIN Major Code: 0X00C2 Minor Code: 100 (0X0064)

**Description**
WINREGISTERCLASS PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINREGISTERCLASS

**Minor Code**
100 (0X0064)
PMWIN Major Code: 0X00C2 Minor Code: 101 (0X0065)

Description  
WINDEFWINDOWPROC PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINDEFWINDOWPROC

Minor Code  
101 (0X0065)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 102 (0X0066)

Description  
WINDESTROYWINDOW PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYWINDOW

Minor Code  
102 (0X0066)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A
PMWIN Major Code: 0X00C2 Minor Code: 103 (0X0067)

**Description**
WINSHOWWINDOW PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSHOWWINDOW

**Minor Code**
103 (0X0067)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
FSHOW = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 104 (0X0068)

**Description**
WINQUERYWINDOWRECT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWRECT

**Minor Code**
104 (0X0068)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
RCLDEST = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 105 (0X0069)

**Description**
WINCREATEWINDOW PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINCREATEWINDOW

Minor Code: 105 (0X0069)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%RESPARAMS = %A, PCTLDATA = %A, ID = %W
HWINDBEHIND = %A, HWNDOWNER = %A, CY = %W
CX = %W, Y = %W, X = %W, FLSTYLE = %D
PSZNAME = %A, PSZCLASS = %A, HWNDPARENT = %A
%CB = %W, FLCREATEFLAGS = %D
HMODRESOURCES = %W, IDRESOURCES = %W
PSZNAME -> %S
PSZCLASS -> %S

PMWIN Major Code: 0X00C2 Minor Code: 106 (0X006A)

Description: WINCREATESTDWINDOW PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINCREATESTDWINDOW

Minor Code: 106 (0X006A)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%HWINDCOMPONENT = %A, IDRESOURCES = %W, HMOD = %W
STYLECLIENT = %D, PSZTITLE = %A, PSZCLIENTCLASS = %A
FLCREATEFLAGS = %A, FLSTYLE = %D, HWNDPARENT = %A
%HWINDCOMPONENT -> %A
%FLCREATEFLAGS -> %D
PSZCLIENTCLASS -> %S
PSZTITLE -> %S
PMWIN Major Code: 0X00C2 Minor Code: 107 (0X006B)

**Description**
WINENABLEWINDOW PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINENABLEWINDOW

**Minor Code**
107 (0X006B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%FENABLE = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 108 (0X006C)

**Description**
WINISWINDOWENABLED PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOWENABLED

**Minor Code**
108 (0X006C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 109 (0X006D)

**Description**
WINISWINDOWVISIBLE PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOWVISIBLE
PMWIN Major Code: 0X00C2 Minor Code: 109 (0X006D)

Description
WINQUERYWINDOWTEXT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWTEXT

PMWIN Major Code: 0X00C2 Minor Code: 110 (0X006E)

Description
WINQUERYWINDOWTEXT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWTEXT

PMWIN Major Code: 0X00C2 Minor Code: 111 (0X006F)

Description
WINSETWINDOWTEXT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWTEXT

Traced Parameters

%HWND = %A
PMWIN Major Code: 0X00C2 Minor Code: 112 (0X0070)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWTEXTLENGTH</td>
</tr>
<tr>
<td>Minor Code</td>
<td>112 (0X0070)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%HWND = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 113 (0X0071)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINWINDOWFROMID PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMID</td>
</tr>
<tr>
<td>Minor Code</td>
<td>113 (0X0071)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%ID = %W, HWNDPARENT = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 114 (0X0072)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINMULTWINDOWFROMIDS PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINMULTWINDOWFROMIDS</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Minor Code</td>
<td>114 (0X0072)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
| Traced Parameters | %IDLAST = %W, IDFIRST = %W  
|                 | PRGHWND = %A, HWNDPARENT = %A                   |

PMWIN Major Code: 0X00C2 Minor Code: 115 (0X0073)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINISWINDOW PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>115 (0X0073)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%HWND = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 116 (0X0074)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYWINDOW PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>116 (0X0074)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
Traced Parameters

%FLOCK = %W, CMD = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 117 (0X0075)

Description
WINISCHILD PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINISCHILD

Minor Code
117 (0X0075)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDPARENT = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 118 (0X0076)

Description
WINQUERYWINDOWPROCESS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWPROCESS

Minor Code
118 (0X0076)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%TID = %A, PPID = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 119 (0X0077)
WINQUERYOBJECTWINDOW PRE-INVOCATION

Description
WINQUERYOBJECTWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYOBJECTWINDOW

Minor Code
119 (0X0077)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 120 (0X0078)

WINQUERYDESKTOPWINDOW PRE-INVOCATION

Description
WINQUERYDESKTOPWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDESKTOPWINDOW

Minor Code
120 (0X0078)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HDC = %A

PMWIN Major Code: 0X00C2 Minor Code: 121 (0X0079)

WINSUBCLASSWINDOW PRE-INVOCATION

Description
WINSUBCLASSWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSUBCLASSWINDOW

Minor Code
121 (0X0079)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

\%FNWP = \%A, HWND = \%A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 122 (0X007A)

Description
WINQUERYCLASSNAME PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLASSNAME

Minor Code
122 (0X007A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%CH = \%A, CCHMAX = \%W, HWND = \%A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 123 (0X007B)

Description
WINQUERYCLASSINFO PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLASSINFO

Minor Code
123 (0X007B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%CLASSINFO = \%A, PSZCLASSNAME = \%A

PSZCLASSNAME -> \%S

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 124 (0X007C)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYACTIVEWINDOW PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYACTIVEWINDOW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>124 (0X007C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%FLOCK = %W, HWNDDESKTOP = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 125 (0X007D)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYSYSMODALWINDOW PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSMODALWINDOW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>125 (0X007D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%FLOCK = %W, HWNDDESKTOP = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 126 (0X007E)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINSETSYSMODALWINDOW PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSMODALWINDOW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>126 (0X007E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%FLOCK = %W, HWNDDESKTOP = %A</td>
</tr>
</tbody>
</table>
Trace Types

No types assigned.

Traced Parameters

%HWND = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 127 (0X007F)

Description

WINLOCKWINDOW PRE-INVOCATION

Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINLOCKWINDOW

Minor Code

127 (0X007F)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%FLOCK = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 128 (0X0080)

Description

WINREGISTERWINDOWDESTROY PRE-INVOCATION

Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINREGISTERWINDOWDESTROY

Minor Code

128 (0X0080)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%FREGISTER = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 129 (0X0081)
WINQUERYWINDOWLOCKCOUNT PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWLOCKCOUNT

Minor Code

129 (0X0081)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 130 (0X0082)

WINQUERYWINDOWUSHORT PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWUSHORT

Minor Code

130 (0X0082)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%INDEX = %W, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 131 (0X0083)

WINSETWINDOWUSHORT PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWUSHORT

Minor Code

131 (0X0083)

Trace Groups

No groups assigned.
Trace Types: No types assigned.

Traced Parameters:

%US = %W, INDEX = %W, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 132 (0X0084)

Description: WINQUERYWINDOWULONG/WINQUERYWINDOWPTR PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWULONG
Minor Code: 132 (0X0084)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%INDEX = %W, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 133 (0X0085)

Description: WINSETWINDOWULONG/WINSETWINDOWPTR PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWULONG
Minor Code: 133 (0X0085)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%UL = %D, INDEX = %W, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 134 (0X0086)
Description	WINSETWINDOWBITS PRE-INVOCATION

Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWBITS

Minor Code	134 (0X0086)

Trace Groups	No groups assigned.

Trace Types	No types assigned.

Traced Parameters

%FLMASK = %D, FLDATA = %D, INDEX = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 135 (0X0087)

Description	WINCANCELSHUTDOWN PRE-INVOCATION

Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINCANCELSHUTDOWN

Minor Code	135 (0X0087)

Trace Groups	No groups assigned.

Trace Types	No types assigned.

Traced Parameters

%FCANCELALWAYS = %W, HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 150 (0X0096)

Description	WINLISTBOXWNDPROC PRE-INVOCATION

Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINLISTBOXWNDPROC

Minor Code	150 (0X0096)

Trace Groups	No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 151 (0X0097)

Description
WINDEFQUEUEPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDEFQUEUEPROC

Minor Code
151 (0X0097)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 152 (0X0098)

Description
WINDESKTOPWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDESKTOPWNDPROC

Minor Code
152 (0X0098)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 153 (0X0099)
WINQUEUEWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUEUEWNDPROC

Minor Code
153 (0X0099)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 154 (0X009A)

WINSETQUEUEPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETQUEUEPROC

Minor Code
154 (0X009A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FNQUEUEPROC = %A

PMWIN Major Code: 0X00C2 Minor Code: 156 (0X009C)

WINDOWUPDATERECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDOWUPDATERECT

Minor Code
156 (0X009C)

Trace Groups
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%FSYNCUPDATE = %W, FS = %W, PRCL = %A, PWND = %W

PMWIN Major Code: 0X00C2 Minor Code: 157 (0X009D)

Description
WINSAVEWINDOWPOS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSAVEWINDOWPOS

Minor Code
157 (0X009D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CSWP = %W, LPSWP = %A, HSVWP = %W

PMWIN Major Code: 0X00C2 Minor Code: 158 (0X009E)

Description
WINSYSTEMSHUTDOWN PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSYSTEMSHUTDOWN

Minor Code
158 (0X009E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
NO PARAMETERS

PMWIN Major Code: 0X00C2 Minor Code: 162 (0X00A2)
COPYWINDOWRECT PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.COPYWINDOWRECT

Minor Code: 162 (0x00A2)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%WRC = %A, PWND = %W

PMWIN Major Code: 0x00C2 Minor Code: 163 (0x00A3)

FINDTOPWINDOW PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.FINDTOPWINDOW

Minor Code: 163 (0x00A3)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%WND = %W, PSWP = %A, CSWP = %W

PMWIN Major Code: 0x00C2 Minor Code: 200 (0x00C8)

WINREGISTERCLASS POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WinRegisterClass

Minor Code: 200 (0x00C8)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 201 (0X00C9)

Description  WINDEFWINDOWPROC POST-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WinDefWindowProc
Minor Code  201 (0X00C9)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  

---

PMWIN Major Code: 0X00C2 Minor Code: 202 (0X00CA)

Description  WINDESTROYWINDOW POST-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYWINDOW
Minor Code  202 (0X00CA)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  

---

PMWIN Major Code: 0X00C2 Minor Code: 203 (0X00CB)
WINSHOWWINDOW POST-INVOCATION

Description: WINSHOWWINDOW POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinShowWindow

Minor Code: 203 (0X00CB)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 204 (0X00CC)

Description: WINQUERYWINDOWRECT POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWRECT

Minor Code: 204 (0X00CC)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 205 (0X00CD)

Description: WINBEGINPAINT POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINBEGINPAINT

Minor Code: 205 (0X00CD)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 206 (0X00CE)

Description
WINENDPAINT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINENDPAINT

Minor Code
206 (0X00CE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 207 (0X00CF)

Description
WINGETPS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETPS

Minor Code
207 (0X00CF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 208 (0X00D0)
| Description       | WINGETCLIPPS POST-INVOCATION
|-------------------|----------------------------------
| Tracepoint        | Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETCLIPPS
| Minor Code        | 208 (0X00D0)
| Trace Groups      | No groups assigned.
| Trace Types       | No types assigned.
| Traced Parameters | DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 209 (0X00D1)

| Description       | WINRELEASEPS POST-INVOCATION
|-------------------|----------------------------------
| Tracepoint        | Public symbol defined dynamic tracepoint: PMWIN.POSTWINRELEASEPS
| Minor Code        | 209 (0X00D1)
| Trace Groups      | No groups assigned.
| Trace Types       | No types assigned.
| Traced Parameters | DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 210 (0X00D2)

| Description       | WINOPENWINDOWDC POST-INVOCATION
|-------------------|----------------------------------
| Tracepoint        | Public symbol defined dynamic tracepoint: PMWIN.POSTWINOPENWINDOWDC
| Minor Code        | 210 (0X00D2)
| Trace Groups      | No groups assigned. 
PMWIN Major Code: 0X00C2 Minor Code: 211 (0X00D3)

**Description**  
WINSCROLLWINDOW POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WinScrollWindow

**Minor Code**  
211 (0X00D3)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 212 (0X00D4)

**Description**  
WINFILLRECT POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.POSTWINFILLRECT

**Minor Code**  
212 (0X00D4)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 213 (0X00D5)
WINCREATEWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WinCreateWindow

Minor Code: 213 (0X00D5)

No groups assigned.

No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 214 (0X00D6)

WINCREATESTDWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WinCreateStdWindow

Minor Code: 214 (0X00D6)

No groups assigned.

No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 215 (0X00D7)

WINENABLEWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEWINDOW

Minor Code: 215 (0X00D7)

No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 216 (0X00D8)

**Description**
WINISWINDOWENABLED POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOWENABLED

**Minor Code**
216 (0X00D8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 217 (0X00D9)

**Description**
WINENABLEWINDOWUPDATE POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEWINDOWUPDATE

**Minor Code**
217 (0X00D9)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 218 (0X00DA)
WINISWINDOWVISIBLE POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOWVISIBLE

Minor Code
218 (0X00DA)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 219 (0X00DB)

WINQUERYWINDOWTEXT POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWTEXT

Minor Code
219 (0X00DB)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 220 (0X00DC)

WINSETWINDOWTEXT POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWTEXT

Minor Code
220 (0X00DC)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 221 (0X00DD)

Description
WINQUERYWINDOWTEXTLENGTH POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWTEXTLENGTH

Minor Code
221 (0X00DD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 222 (0X00DE)

Description
WINWINDOWFROMID POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMID

Minor Code
222 (0X00DE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 223 (0X00DF)
WINMULTWINDOWFROMIDS POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINMULTWINDOWFROMIDS

Minor Code
223 (0X00DF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 224 (0X00E0)

WINISWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOW

Minor Code
224 (0X00E0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 225 (0X00E1)

WINQUERYWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOW

Minor Code
225 (0X00E1)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 226 (0X00E2)

Description
WINSETPARENT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPARENT

Minor Code
226 (0X00E2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 227 (0X00E3)

Description
WINISCHILD POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINISCHILD

Minor Code
227 (0X00E3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 228 (0X00E4)
WINSETOWNER POST-INVOCATION

Description
WINSETOWNER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETOWNER

Minor Code
228 (0X00E4)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 229 (0X00E5)

Description
WINQUERYWINDOWPROCESS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWPROCESS

Minor Code
229 (0X00E5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 230 (0X00E6)

Description
WINQUERYOBJECTWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYOBJECTWINDOW

Minor Code
230 (0X00E6)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 231 (0X00E7)

Description: WINQUERYDESKTOPWINDOW POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinQueryDesktopWindow
Minor Code: 231 (0X00E7)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 232 (0X00E8)

Description: WINLOADSTRING POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinLoadString
Minor Code: 232 (0X00E8)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 233 (0X00E9)
Description: WINLOADMESSAGE POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinLoadMessage

Minor Code: 233 (0X00E9)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 234 (0X00EA)

Description: WINQUERYVERSION POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinQueryVersion

Minor Code: 234 (0X00EA)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 235 (0X00EB)

Description: WININITIALIZE POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinInitialize

Minor Code: 235 (0X00EB)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 236 (0X00EC)

Description
WINTERMINATE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinTerminate

Minor Code
236 (0X00EC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 300 (0X012C)

Description
WINSETWINDOWPOS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWPOS

Minor Code
300 (0X012C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FS = %W, CY = %W, CX = %W, Y = %W, X = %W
HWNDINSERTBEHIND = %A, HWND = %A
PMWIN Major Code: 0X00C2 Minor Code: 301 (0X012D)

Description
WINQUERYWINDOWPOS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWPOS

Minor Code
301 (0X012D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%SWP = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 302 (0X012E)

Description
WINSETMULTWINDOWPOS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETMULTWINDOWPOS

Minor Code
302 (0X012E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CSWP = %W, PSWP = %A
%SWP->FS = %W, PSWP->CY = %W
PSWP->CX = %W, PSWP->Y = %W, PSWP->X = %W
PSWP->HWNDINSERTBEHIND = %A, PSWP->HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 303 (0X012F)

Description
WINSETPARENT PRE-INVOCATION
PMWIN Major Code: 0X00C2 Minor Code: 303 (0X012F)

Description
WINSETPARENT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETPARENT

Minor Code
303 (0X012F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FREDRAW = %W, HWNDNEWPARENT = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 304 (0X0130)

Description
WINSETOWNER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETOWNER

Minor Code
304 (0X0130)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDNEWOWNER = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 305 (0X0131)

Description
WINUPDATEWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINUPDATEWINDOW

Minor Code
305 (0X0131)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PMWIN Major Code: 0X00C2 Minor Code: 306 (0X0132)

Description
WININVALIDATERECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WININVALIDATERECT

Minor Code
306 (0X0132)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FINCLUDECHILDREN = %W, PWRC = %A, HWND = %A
%WRC->XLEFT = %D, PWRC->YBOTTOM = %D
PWRC->XRIGHT = %D, PWRC->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 307 (0X0133)

Description
WININVALIDATEREGION PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WININVALIDATEREGION

Minor Code
307 (0X0133)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FINCLUDECHILDREN = %W, HRGN = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 308 (0X0134)
**Description**  
WINDRAWTEXT PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINDRAWTEXT

**Minor Code**  
308 (0X0134)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%RGFCMD = %W, CLRBACK = %D, CLRFORE = %D  
PRCL = %A, PCHTEXT = %A, CCHTEXT = %W, HPS = %A  
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D  
PRCL->XRIGHT = %D, PRCL->YTOP = %D  
PCHTEXT -> %S

PMWIN Major Code: 0X00C2 Minor Code: 309 (0X0135)

**Description**  
WINVALIDATERECT PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINVALIDATERECT

**Minor Code**  
309 (0X0135)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%FINCLUDECHILDREN = %W, PRCL = %A, HWND = %A  
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D  
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 310 (0X0136)
WINVALIDATEREGION PRE-INVOCATION

**Description**
Public symbol defined dynamic tracepoint: PMWIN.WINVALIDATEREGION

**Minor Code**
310 (0X0136)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%INCLUDECHILDREN = %W, HRGN = %A, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 311 (0X0137)

WINWINDOWFROMDC PRE-INVOCATION

**Description**
Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMDC

**Minor Code**
311 (0X0137)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HDC = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 312 (0X0138)

WINQUERYWINDOWDC PRE-INVOCATION

**Description**
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWDC

**Minor Code**
312 (0X0138)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 313 (0X0139)

Description
WINGETSCREENPS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETSCREENPS

Minor Code
313 (0X0139)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 314 (0X013A)

Description
WINQUERYUPDATERECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYUPDATERECT

Minor Code
314 (0X013A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%RCL = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 315 (0X013B)
Description: WINQUERYUPDATEREGION PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYUPDATEREGION

Minor Code: 315 (0X013B)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%HRGN = %A, HWND = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 316 (0X013C)

Description: WINEXCLUDEUPDATEREGION PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINEXCLUDEUPDATEREGION

Minor Code: 316 (0X013C)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%HWND = %A, HPS = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 317 (0X013D)

Description: WINLOCKWINDOWUPDATE PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINLOCKWINDOWUPDATE

Minor Code: 317 (0X013D)

Trace Groups: No groups assigned.

Trace Types: No types assigned.
Traced Parameters

%HWNDLOCKUPDATE = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 318 (0X013E)

Description
WINLOCKVISREGIONS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINLOCKVISREGIONS

Minor Code
318 (0X013E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FLOCK = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 319 (0X013F)

Description
WINBEGINPAINT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINBEGINPAINT

Minor Code
319 (0X013F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%RCLPAINT = %A, HPS = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 320 (0X0140)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINENDPAINT PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINENDPAINT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>320 (0X0140)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%HPS = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 321 (0X0141)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETPS PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINGETPS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>321 (0X0141)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%HWND = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 322 (0X0142)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETCLIPPS PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINGETCLIPPS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>322 (0X0142)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
Traced Parameters

%FS = %W, HWNDCLIP = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 323 (0X0143)

Description
WINRELEASEPS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINRELEASEPS

Minor Code
323 (0X0143)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HPS = %A

PMWIN Major Code: 0X00C2 Minor Code: 324 (0X0144)

Description
WINOPENWINDOWDC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINOPENWINDOWDC

Minor Code
324 (0X0144)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 325 (0X0145)
Description: WINSCROLLWINDOW PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSCROLLWINDOW

Minor Code: 325 (0X0145)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%RGFSW = %W, PRCLUPDATE = %A, HRGNUPDATE = %A
PRCLCLIP = %A, PRCLSCROLL = %A, DY = %W, DX = %W
HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 326 (0X0146)

Description: WINFILLRECT PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINFILLRECT

Minor Code: 326 (0X0146)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%LCOLOR = %D, PRCL = %A, HPS = %A
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 327 (0X0147)

Description: WINENABLEWINDOWUPDATE PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINENABLEWINDOWUPDATE
Minor Code: 327 (0X0147)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%FENABLE = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 400 (0X0190)

Description: WINSETWINDOWPOS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinSetWindowPos

Minor Code: 400 (0X0190)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 401 (0X0191)

Description: WINQUERYWINDOWPOS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWPOS

Minor Code: 401 (0X0191)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 402 (0X0192)

Description
WINSETMULTWINDOWPOS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetMultWindowPos

Minor Code
402 (0X0192)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 403 (0X0193)

Description
WINUPDATEWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINUPDATEWINDOW

Minor Code
403 (0X0193)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 404 (0X0194)

Description
WININVALIDATERECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinInvalidateRect
**Minor Code**: 404 (0X0194)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 405 (0X0195)

**Description**: WININVALIDATEREGION POST-INVOCATION

**Tracepoint**: Public symbol defined dynamic tracepoint: PMWIN.WinInvalidateRegion

**Minor Code**: 405 (0X0195)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 406 (0X0196)

**Description**: WININVERTRECT POST-INVOCATION

**Tracepoint**: Public symbol defined dynamic tracepoint: PMWIN.POSTWININVERTRECT

**Minor Code**: 406 (0X0196)

**Trace Groups**: No groups assigned.

**Trace Types**: No types assigned.

**Traced Parameters**

DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 407 (0X0197)

**Description**
WINDRAWBITMAP POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinDrawBitmap

**Minor Code**
407 (0X0197)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

$DX = \%W, AX = \%W$

PMWIN Major Code: 0X00C2 Minor Code: 408 (0X0198)

**Description**
WINDRAWTEXT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.DRAWTEXT

**Minor Code**
408 (0X0198)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

$DX = \%W, AX = \%W$

PMWIN Major Code: 0X00C2 Minor Code: 409 (0X0199)

**Description**
WINDRAWBORDER POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinDrawBorder
PMWIN Major Code: 0X00C2 Minor Code: 409 (0X0199)

Trace Groups    No groups assigned.
Trace Types      No types assigned.
Traced Parameters DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 410 (0X019A)

Description      WINVALIDATERECT POST-INVOCATION
Tracepoint       Public symbol defined dynamic tracepoint: PMWIN.WinValidateRect
Minor Code       410 (0X019A)
Trace Groups      No groups assigned.
Trace Types       No types assigned.
Traced Parameters DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 411 (0X019B)

Description      WINVALIDATEREGION POST-INVOCATION
Tracepoint       Public symbol defined dynamic tracepoint: PMWIN.WinValidateRegion
Minor Code       411 (0X019B)
Trace Groups      No groups assigned.
Trace Types       No types assigned.
Traced Parameters DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 412 (0X019C)

Description
WINWINDOWFROMDC POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMDC

Minor Code
412 (0X019C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 413 (0X019D)

Description
WINQUERYWINDOWDC POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWDC

Minor Code
413 (0X019D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 414 (0X019E)

Description
WINGETSCREENPS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETSCREENPS
PMWIN Major Code: 0X00C2 Minor Code: 414 (0X019E)

Minor Code: 414 (0X019E)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 415 (0X019F)

Description: WINQUERYUPDATERECT POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYUPDATERECT
Minor Code: 415 (0X019F)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 416 (0X01A0)

Description: WINQUERYUPDATEREGION POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYUPDATEREGION
Minor Code: 416 (0X01A0)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 417 (0X01A1)

Description  
WINEXCLUDEUPDATEREGION POST-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.POSTWINEXCLUDEUPDATEREGION

Minor Code  
417 (0X01A1)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 418 (0X01A2)

Description  
WINLOCKWINDOWUPDATE POST-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOCKWINDOWUPDATE

Minor Code  
418 (0X01A2)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 419 (0X01A3)

Description  
WINLOCKVISREGIONS POST-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WinLockVisRegions
PMWIN Major Code: 0X00C2 Minor Code: 500 (0X01F4)

Description
WINBEGINENUMWINDOWS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINBEGINENUMWINDOWS

Minor Code
500 (0X01F4)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 501 (0X01F5)

Description
WINGETNEXTWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETNEXTWINDOW

Minor Code
501 (0X01F5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HENUM = %A
PMWIN Major Code: 0X00C2 Minor Code: 502 (0X01F6)

Description  
WINENDENUMWINDOWS PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINENDENUMWINDOWS

Minor Code  
502 (0X01F6)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

%HENUM = %A

PMWIN Major Code: 0X00C2 Minor Code: 503 (0X01F7)

Description  
WINWINDOWFROMPOINT PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMPOINT

Minor Code  
503 (0X01F7)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

%FLOCK = %W, FCHILDREN = %W
PPTL = %A, HWND = %A
%TL->X = %D, PPTL->Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 504 (0X01F8)

Description  
WINMAPWINDOWPOINTS PRE-INVOCATION
Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINMAPWINDOWPOINTS

Minor Code
504 (0X01F8)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CWPT = %W, PRGPTL = %A, HWNDTO = %A, HWNDFROM = %A
%RGPTL->X = %D, PRGPTL->Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 600 (0X0258)

Description
WINSETACTIVEWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETACTIVEWINDOW

Minor Code
600 (0X0258)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 601 (0X0259)

Description
WINSUBCLASSWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSUBCLASSWINDOW

Minor Code
601 (0X0259)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
PMWIN Major Code: 0X00C2 Minor Code: 602 (0X025A)

Description
WINQUERYCLASSNAME POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLASSNAME

Minor Code
602 (0X025A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 603 (0X025B)

Description
WINQUERYCLASSINFO POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinQueryClassInfo

Minor Code
603 (0X025B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 604 (0X025C)
WINQUERYACTIVEWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYACTIVEWINDOW

Minor Code
604 (0X025C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 605 (0X025D)

WINISTHREADACTIVE POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINISTHREADACTIVE

Minor Code
605 (0X025D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 606 (0X025E)

WINQUERYSYSMODALWINDOW POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSMODALWINDOW

Minor Code
606 (0X025E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
PMWIN Major Code: 0X00C2 Minor Code: 607 (0X025F)

**Description**
WINSETSYSMODALWINDOW POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETSYSMODALWINDOW

**Minor Code**
607 (0X025F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 608 (0X0260)

**Description**
WINLOCKWINDOW POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinLockWindow

**Minor Code**
608 (0X0260)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 609 (0X0261)
Description: WINREGISTERWINDOWDESTROY POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINREGISTERWINDOWDESTROY

Minor Code: 609 (0X0261)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 610 (0X0262)

Description: WINQUERYWINDOWLOCKCOUNT POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWLOCKCOUNT

Minor Code: 610 (0X0262)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 611 (0X0263)

Description: WINQUERYWINDOWUSHORT POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWVALUE

Minor Code: 611 (0X0263)

Trace Groups: No groups assigned.

Trace Types: No types assigned.
PMWIN Major Code: 0X00C2 Minor Code: 612 (0X0264)

**Description**  
WINSETWINDOWUSHORT POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWVALUE

**Minor Code**  
612 (0X0264)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 613 (0X0265)

**Description**  
WINQUERYWINDOWUSHORT/WINQUERYWINDOWULONG/WINQUERYWINDOWPTR POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWVALUE

**Minor Code**  
613 (0X0265)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 614 (0X0266)
**WINSETWINDOWUSHORT/WINSETWINDOWULONG/WINSETWINDOWPTR POST-INVOCATION**

**Description**
WINSETWINDOWUSHORT/WINSETWINDOWULONG/WINSETWINDOWPTR POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWVALUE

**Minor Code**
614 (0x0266)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{DX} = \%W, \text{AX} = \%W
\]

--------------------------------------------

**PMWIN Major Code: 0x00C2 Minor Code: 615 (0x0267)**

**Description**
WINSETWINDOWBITS POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWBITS

**Minor Code**
615 (0x0267)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\text{DX} = \%W, \text{AX} = \%W
\]

--------------------------------------------

**PMWIN Major Code: 0x00C2 Minor Code: 650 (0x028A)**

**Description**
WINBEGINENUMWINDOWS POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINBEGINENUMWINDOWS

**Minor Code**
650 (0x028A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
PMWIN Major Code: 0X00C2 Minor Code: 651 (0X028B)

Description
WINGETNEXTWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETNEXTWINDOW

Minor Code
651 (0X028B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 652 (0X028C)

Description
WINENDENUMWINDOWS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINENDENUMWINDOWS

Minor Code
652 (0X028C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 653 (0X028D)
Description: WINWINDOWFROMPOINT POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMPOINT

Minor Code: 653 (0X028D)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

  DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 654 (0X028E)

Description: WINMAPWINDOWPOINTS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAPWINDOWPOINTS

Minor Code: 654 (0X028E)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

  DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 700 (0X02BC)

Description: WINQUERYQUEUEINFO PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYQUEUEINFO

Minor Code: 700 (0X02BC)

Trace Groups: No groups assigned.

Trace Types: No types assigned.
Traced Parameters

%CBCOPY = %W, PMQI = %A, HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 701 (0X02BD)

Description
WINCREATEMSGQUEUE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCREATEMSGQUEUE

Minor Code
701 (0X02BD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CMSG = %W

PMWIN Major Code: 0X00C2 Minor Code: 702 (0X02BE)

Description
WINDESTRUCTIONQUEUE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDESTRUCTIONQUEUE

Minor Code
702 (0X02BE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 703 (0X02BF)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETMSG PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINGETMSG</td>
</tr>
<tr>
<td>Minor Code</td>
<td>703 (0X02BF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
| Traced Parameters | %MSGFILTERLAST = %W, MSGFILTERFIRST = %W  
HWNDFILTER = %A, PQMSG = %A |

PMWIN Major Code: 0X00C2 Minor Code: 704 (0X02C0)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINPEEKMSG PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINPEEKMSG</td>
</tr>
<tr>
<td>Minor Code</td>
<td>704 (0X02C0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
| Traced Parameters | %FS = %W, MSGFILTERLAST = %W, MSGFILTERFIRST = %W  
HWNDFILTER = %A, PQMSG = %A |

PMWIN Major Code: 0X00C2 Minor Code: 705 (0X02C1)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINDISPATCHMSG PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINDISPATCHMSG</td>
</tr>
<tr>
<td>Minor Code</td>
<td>705 (0X02C1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td></td>
</tr>
<tr>
<td>Trace Types</td>
<td></td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 706 (0X02C2)

Description
WINPOSTMSG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINPOSTMSG

Minor Code
706 (0X02C2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%QMSG = %A
%QMSG->HWND = %A, PQMSG->MSG = %W
PQMSG->MP1 = %D, PQMSG->MP2 = %D, PQMSG->TIME = %D
PQMSG->PTL.X = %D, PQMSG->PTL.Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 707 (0X02C3)

Description
WININSENDMSG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WININSENDMSG

Minor Code
707 (0X02C3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

Erica K. Petersen
PMWIN Major Code: 0X00C2 Minor Code: 708 (0X02C4)

**Description**
WINBROADCASTMSG PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINBROADCASTMSG

**Minor Code**
708 (0X02C4)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%RGF = %W, MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 709 (0X02C5)

**Description**
WINWAITMSG PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINWAITMSG

**Minor Code**
709 (0X02C5)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%MSGLAST = %W, MSGFIRST = %W

PMWIN Major Code: 0X00C2 Minor Code: 710 (0X02C6)

**Description**
WINQUERYQUEUESTATUS PRE-INVOCATION
Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYQUEUESTATUS

Minor Code
710 (0X02C6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 711 (0X02C7)

Description
WINPOSTQUEUEMSG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINPOSTQUEUEMSG

Minor Code
711 (0X02C7)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HMQ = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 712 (0X02C8)

Description
WINQUERYMSGPOS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYMSGPOS

Minor Code
712 (0X02C8)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HMQ = %A
PMWIN Major Code: 0X00C2 Minor Code: 713 (0X02C9)

Description: WINQUERYMSGTIME PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYMSGTIME
Minor Code: 713 (0X02C9)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 714 (0X02CA)

Description: WINMSGSEMWAIT PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINMSGSEMWAIT
Minor Code: 714 (0X02CA)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%DTTIMEOUT = %D, HSEM = %A

PMWIN Major Code: 0X00C2 Minor Code: 715 (0X02CB)

Description: WINMSGMUXSEMWAIT PRE-INVOCATION
Tracepoint: 0X00C2 Minor Code: 715 (0X02CB)

Description: WINSETMSGINTEREST PRE-INVOCATION

Tracepoint: 0X00C2 Minor Code: 717 (0X02CD)

Description: WINSENDMSG PRE-INVOCATION
Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 751 (0X02EF)

Description
WINSENDQUEUEMSG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSENDQUEUEMSG

Minor Code
751 (0X02EF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 752 (0X02F0)

Description
CREATEQUEUE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.CREATEQUEUE

Minor Code
752 (0X02F0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CMSGS = %W

PMWIN Major Code: 0X00C2 Minor Code: 753 (0X02F1)
**Description**
FREEQUEUE PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.FREEQUEUE

**Minor Code**
753 (0X02F1)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%SMQ = %W

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 754 (0X02F2)**

**Description**
CLEARQUEUEGLOBALS PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.CLEARQUEUEGLOBALS

**Minor Code**
754 (0X02F2)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%SMQ = %W

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 755 (0X02F3)**

**Description**
REASSIGNINPUT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.REASSIGNINPUT

**Minor Code**
755 (0X02F3)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

%SMQ = %W

PMWIN Major Code: 0X00C2 Minor Code: 756 (0X02F4)

Description
ASSOCIATEQUEUE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.ASSOCIATEQUEUE

Minor Code
756 (0X02F4)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%SMQ = %W

PMWIN Major Code: 0X00C2 Minor Code: 757 (0X02F5)

Description
READMESSAGE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.READMESSAGE

Minor Code
757 (0X02F5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FREMOVEMSG = %W, MSGMAXFILTER = %W, MSGMINFILTER = %W
PWNDFILTER = %W, LPMRG = %A, SMQ = %W

PMWIN Major Code: 0X00C2 Minor Code: 800 (0X0320)
WINQUERYQUEUEINFO POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYQUEUEINFO

Minor Code
800 (0X0320)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 801 (0X0321)

WINCREATESMSGQUEUE POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WinCreateMsgQueue

Minor Code
801 (0X0321)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 802 (0X0322)

WINDESTRYMMSGQUEUE POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WinDestroyMsgQueue

Minor Code
802 (0X0322)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 803 (0X0323)

Description
WINCANCELSHUTDOWN POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinCancelShutdown

Minor Code
803 (0X0323)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 804 (0X0324)

Description
WINGETMSG/WINPEEKMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMSG

Minor Code
804 (0X0324)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 805 (0X0325)
Description: WINPEEKMSG POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINPEEKMSG

Minor Code: 805 (0X0325)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 806 (0X0326)

Description: WINDISPATCHMSG POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINDISPATCHMSG

Minor Code: 806 (0X0326)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 807 (0X0327)

Description: WINPOSTMSG/WINPOSTQUEUEMSG POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINPOSTMSG

Minor Code: 807 (0X0327)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 808 (0X0328)

Description
WININSENDMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWININSENDMSG

Minor Code
808 (0X0328)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 809 (0X0329)

Description
WINBROADCASTMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINBROADCASTMSG

Minor Code
809 (0X0329)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 810 (0X032A)
WINWAITMSG POST-INVOCATION

Description
WINWAITMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINWAITMSG

Minor Code
810 (0X032A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 811 (0X032B)

WINQUERYQUEUESTATUS POST-INVOCATION

Description
WINQUERYQUEUESTATUS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYQUEUESTATUS

Minor Code
811 (0X032B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 812 (0X032C)

WINPOSTQUEUEMSG POST-INVOCATION

Description
WINPOSTQUEUEMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINPOSTQUEUEMSG

Minor Code
812 (0X032C)

Trace Groups
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 813 (0X032D)

Description
WINQUERYMSGPOS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYMSGPOS

Minor Code
813 (0X032D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 814 (0X032E)

Description
WINQUERYMSGTIME POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYMSGTIME

Minor Code
814 (0X032E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 815 (0X032F)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINMSGSEMWAIT POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINMSGSEMWAIT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>815 (0X032F)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 816 (0X0330)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINMSGSEMWAIT/WINMSGMUXSEMWAIT POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINMSGMUXSEMWAIT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>816 (0X0330)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 817 (0X0331)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINSETMSGINTEREST POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WinSetMsgInterest</td>
</tr>
<tr>
<td>Minor Code</td>
<td>817 (0X0331)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 818 (0X0332)

Description
WINSENDMSG/WINSENDQUEUEMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSENDMSG

Minor Code
818 (0X0332)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 900 (0X0384)

Description
WINSETFOCUS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETFOCUS

Minor Code
900 (0X0384)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HWNDSETFOCUS = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 901 (0X0385)
WINFOCUSCHANGE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINFOCUSCHANGE

Minor Code
901 (0X0385)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FSFOCUSCHANGE = %W, HWNDSETFOCUS = %A
HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 902 (0X0386)

WINSETCAPTURE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETCAPTURE

Minor Code
902 (0X0386)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWND = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 903 (0X0387)

WINQUERYCAPTURE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCAPTURE

Minor Code
903 (0X0387)
PMWIN Major Code: 0X00C2 Minor Code: 904 (0X0388)

Description  WINQUERYFOCUS PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINQUERYFOCUS
Minor Code  904 (0X0388)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

  %FLOCK = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 905 (0X0389)

Description  WINSETACTIVEWINDOW PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINSETACTIVEWINDOW
Minor Code  905 (0X0389)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

  %HWND = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 904 (0X0388)
PMWIN Major Code: 0X00C2 Minor Code: 906 (0X038A)

Description
WINISTHREADACTIVE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINISTHREADACTIVE

Minor Code
906 (0X038A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 907 (0X038B)

Description
WINGETKEYSTATE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETKEYSTATE

Minor Code
907 (0X038B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%VKEY = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 908 (0X038C)

Description
WINGETPHYSKEYSTATE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETPHYSKEYSTATE

Minor Code
908 (0X038C)
Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

%SC = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 909 (0X038D)

Description  
WINENABLEPHYSINPUT PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINENABLEPHYSINPUT

Minor Code  
909 (0X038D)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

%ENABLE = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 910 (0X038E)

Description  
WINISPHYSINPUTENABLED PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINISPHYSINPUTENABLED

Minor Code  
910 (0X038E)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  

%HWNDDESKTOP = %A
PMWIN Major Code: 0x00C2 Minor Code: 911 (0x038F)

**Description**
WINSETKEYBOARDSTATETABLE PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSETKEYBOARDSTATETABLE

**Minor Code**
911 (0x038F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%FSET = %W, PKEYSTATETABLE = %A, HWNDDESKTOP = %A
%KEYSTATETABLE -> %B

---------------------------

PMWIN Major Code: 0x00C2 Minor Code: 912 (0x0390)

**Description**
WINTRACKRECT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINTRACKRECT

**Minor Code**
912 (0x0390)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%TI = %A, HPS = %A, HWND = %A
%TI->CXBORDER = %W, PTI->CYBORDER = %W
PTI->CXGRID = %W, PTI->CYGRID = %W
PTI->CXKEYBOARD = %W, PTI->CYKEYBOARD = %W
PTI->RCLTRACK = %A, PTI->RCLBOUNDARY = %A
PTI->PTLMINTRACKSIZE = %A, PTI->PTLMAXTRACKSIZE = %A
PTI->FS = %W

---------------------------
PMWIN Major Code: 0X00C2 Minor Code: 913 (0X0391)

**Description**
WINSHOWTRACKRECT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSHOWTRACKRECT

**Minor Code**
913 (0X0391)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%FSIZE = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1000 (0X03E8)

**Description**
WINSETFOCUS POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETFOCUS

**Minor Code**
1000 (0X03E8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1001 (0X03E9)

**Description**
WINFOCUSCHANGE POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINFOCUSCHANGE

**Minor Code**
PMWIN Major Code: 0X00C2 Minor Code: 1002 (0X03EA)

Description: WINSETCAPTURE POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETCAPTURE
Minor Code: 1002 (0X03EA)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

\[ DX = %W, AX = %W \]

PMWIN Major Code: 0X00C2 Minor Code: 1003 (0X03EB)

Description: WINQUERYCAPTURE POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCAPTURE
Minor Code: 1003 (0X03EB)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

\[ DX = %W, AX = %W \]
PMWIN Major Code: 0X00C2 Minor Code: 1004 (0X03EC)

Description
WINQUERYFOCUS/WINQUERYSYSTEMMODALWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYFOCUS

Minor Code
1004 (0X03EC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1005 (0X03ED)

Description
WINGETKEYSTATE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETKEYSTATE

Minor Code
1005 (0X03ED)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1006 (0X03EE)

Description
WINGETPHYSKEYSTATE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETPHYSKEYSTATE
PMWIN Major Code: 0X00C2 Minor Code: 1006 (0X03EE)

Description
WINENABLEPHYSINPUT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEPHYSINPUT

Minor Code
1006 (0X03EE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1007 (0X03EF)

Description
WINENABLEPHYSINPUT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEPHYSINPUT

Minor Code
1007 (0X03EF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1008 (0X03F0)

Description
WINISPHYSINPUTENABLED POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINISPHYSINPUTENABLED

Minor Code
1008 (0X03F0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 1009 (0X03F1)

Description
WINSETKEYBOARDSTATETABLE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETKEYBOARDSTATETABLE

Minor Code
1009 (0X03F1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1100 (0X044C)

Description
WINLOADDLG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINLOADDLG

Minor Code
1100 (0X044C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CREATEPARAMS = %A, IDDLG = %W, HMOD = %W
PFNDLGPLPROC = %A, HWDOWNER = %A, HWDPPARENT = %A

PMWIN Major Code: 0X00C2 Minor Code: 1101 (0X044D)

Description
WINDLGBOX PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDLGBOX

**Minor Code**
1101 (0X044D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
- %CREATEPARAMS = %A, IDDLG = %W, HMOD = %W
- PFNDLGPROC = %A, HWNDOWNER = %A, HWNDPARENT = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1102 (0X044E)

**Description**
WINDISMISSDLG PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINDISMISSDLG

**Minor Code**
1102 (0X044E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
- %USRESULT = %W, HWNDDLG = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1103 (0X044F)

**Description**
WINSETDLGITEMSHORT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSETDLGITEMSHORT

**Minor Code**
1103 (0X044F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
PMWIN Major Code: 0X00C2 Minor Code: 1104 (0X0450)

Description
WINQUERYDLGITEMSHORT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDLGITEMSHORT

Minor Code
1104 (0X0450)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FSIGNED = %W, USVALUE = %W, IDITEM = %W, HWNDDLG = %A

PMWIN Major Code: 0X00C2 Minor Code: 1105 (0X0451)

Description
WINSETDLGITEMTEXT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETDLGITEMTEXT

Minor Code
1105 (0X0451)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%SZTEXT = %A, IDITEM = %W, HWNDDLG = %A

PMWIN Major Code: 0X00C2 Minor Code: 1106 (0X0452)

Description
WINQUERYDLGITEMTEXT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDLGITEMTEXT

Minor Code
1106 (0X0452)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CHBUFFER = %A, CCHBUFFERMAX = %W
IDITEM = %W, HWNDDLG = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1107 (0X0453)

Description
WINDEFDLGPLPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDEFDLGPLPROC

Minor Code
1107 (0X0453)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWNDDLG = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1108 (0X0454)

Description
WINALARM PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINALARM

Minor Code
1108 (0X0454)

Trace Groups
No groups assigned.

Trace Types
Traced Parameters

%RGFTYPE = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 1109 (0X0455)

Description
WINMESSAGEBOX PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINMESSAGEBOX

Minor Code
1109 (0X0455)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FLSTYLE = %W, IDWINDOW = %W, PSZCAPTION = %A
PSZTEXT = %A, HWNDOWNER = %A, HWNDPARENT = %A
PSZCAPTION -> %S
PSZTEXT -> %S

PMWIN Major Code: 0X00C2 Minor Code: 1110 (0X0456)

Description
WINPROCESSDLG PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINPROCESSDLG

Minor Code
1110 (0X0456)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDDLG = %A
PMWIN Major Code: 0X00C2 Minor Code: 1111 (0X0457)

Description: WINSENDLGITEMMMSG PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSENDLGITEMMMSG
Minor Code: 1111 (0X0457)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%MP2 = %D, MP1 = %D, MSG = %W
IDITEM = %W, HWNDDL = %A

PMWIN Major Code: 0X00C2 Minor Code: 1112 (0X0458)

Description: WINMAPDLGPOINTS PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINMAPDLGPOINTS
Minor Code: 1112 (0X0458)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%FCALCWINDOWCOORDS = %W, CWPT = %W
PRGWPTL = %A, HWNDDL = %A
%RGWPTL->X = %D, PRGWPTL->Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 1113 (0X0459)
Description  WINSUBSTITUTESTRINGS PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINSUBSTITUTESTRINGS
Minor Code  1113 (0X0459)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%SZDST = %A, CCHDSTMAX = %W, PSZSRC = %A, HWND = %A
PSZSRC -> %S

PMWIN Major Code: 0X00C2 Minor Code: 1114 (0X045A)

Description  WINENUMDLGITEM PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINENUMDLGITEM
Minor Code  1114 (0X045A)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%FLOCK = %W, CODE = %W, HWND = %A, HWNDDLG = %A

PMWIN Major Code: 0X00C2 Minor Code: 1115 (0X045B)

Description  WINCREATEDLG PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINCREATEDLG
Minor Code  1115 (0X045B)
Trace Groups  No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%CREATEPARAMS = %A, PDLGT = %A, PFNDLGPROC = %A
HWNDOWNER = %A, HWNDPARENT = %A
%DLGT->CBTEMPLATE = %W, PDLGT->TYPE = %W
PDLGT->CODEPAGE = %W, PDLGT->OFFADLGTI = %W
PDLGT->FSTEMPLETSTATUS = %W, PDLGT->ITEMFOCUS = %W
PDLGT->COFFPRESPARAMS = %W, PDLGT->ADLGTI[] = %A

PMWIN Major Code: 0X00C2 Minor Code: 1150 (0X047E)

Description
WINEDITWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINEDITWNDPROC

Minor Code
1150 (0X047E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1200 (0X04B0)

Description
WINLOADDLG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinLoadDlg

Minor Code
1200 (0X04B0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PMWIN Major Code: 0X00C2 Minor Code: 1201 (0X04B1)

Description: WINDLGBOX POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinDlgBox
Minor Code: 1201 (0X04B1)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1202 (0X04B2)

Description: WINDISMISSDLG POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinDismissDlg
Minor Code: 1202 (0X04B2)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1203 (0X04B3)

Description:
WINSETDLGITEMSHORT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETDLGITEMSHORT

**Minor Code**
1203 (0X04B3)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = %W, AX = %W \]

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1204 (0X04B4)

**Description**
WINQUERYDLGITEMSHORT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYDLGITEMSHORT

**Minor Code**
1204 (0X04B4)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = %W, AX = %W \]

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1205 (0X04B5)

**Description**
WINSETDLGITEMTEXT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinSetDlgItemText

**Minor Code**
1205 (0X04B5)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1206 (0X04B6)

Description
WINQUERYDLGITEMTEXT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinQueryDlgItemText

Minor Code
1206 (0X04B6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1207 (0X04B7)

Description
WINDEFDLGPROC POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinDefDlgProc

Minor Code
1207 (0X04B7)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1208 (0X04B8)
WINALARM POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINALARM

Minor Code
1208 (0X04B8)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1209 (0X04B9)

WINMESSAGEBOX POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinMessageBox

Minor Code
1209 (0X04B9)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1210 (0X04BA)

WINPROCESSDLG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinProcessDlg

Minor Code
1210 (0X04BA)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

`DX = %W, AX = %W`

PMWIN Major Code: 0X00C2 Minor Code: 1211 (0X04BB)

Description
WINSENDLGITEMMSG POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSENDLGITEMMSG

Minor Code
1211 (0X04BB)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

`DX = %W, AX = %W`

PMWIN Major Code: 0X00C2 Minor Code: 1212 (0X04BC)

Description
WINMAPDLGPOINTS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAPDLGPOINTS

Minor Code
1212 (0X04BC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

`DX = %W, AX = %W`

PMWIN Major Code: 0X00C2 Minor Code: 1213 (0X04BD)
Description: WINSUBSTITUTESTRINGS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINSUBSTITUTESTRINGS

Minor Code: 1213 (0x04bd)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0x00c2 Minor Code: 1214 (0x04be)

Description: WINENUMDLGITM POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINENUMDLGITM

Minor Code: 1214 (0x04be)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

PMWIN Major Code: 0x00c2 Minor Code: 1215 (0x04bf)

Description: WINCREATEDLG POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinCreateDlg

Minor Code: 1215 (0x04bf)

Trace Groups: No groups assigned.

Trace Types: No types assigned.
Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1300 (0X514)

Description
WINLOADMENU PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINLOADMENU

Minor Code
1300 (0X0514)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%IDMENU = %W, HMOD = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1301 (0X515)

Description
WINCREATEMENU PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCREATEMENU

Minor Code
1301 (0X0515)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%LPM = %A, HWNDPARENT = %A

PMWIN Major Code: 0X00C2 Minor Code: 1302 (0X516)
WINFLASHWINDOW PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINFLASHWINDOW

Minor Code
1302 (0X0516)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FFLASH = %W, HWNDFRAME = %A

PMWIN Major Code: 0X00C2 Minor Code: 1303 (0X0517)

WINCREATEFRAMECONTROLS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCREATEFRAMECONTROLS

Minor Code
1303 (0X0517)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%SZTITLE = %A, PFCDATA = %A
HWNDFRAME = %A
%FCDATA->CB = %W, PFCDATA->FLCREATEFLAGS = %D
PFCDATA->HMODRESOURCES = %W, PFCDATA->IDRESOURCES = %W
PSZTITLE -> %S

PMWIN Major Code: 0X00C2 Minor Code: 1304 (0X0518)

WINFORMATFRAME PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINFORMATFRAME
PMWIN Major Code: 0X00C2 Minor Code: 1305 (0X0519)

Description
WINCALCFRAMERECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCALCFRAMERECT

Minor Code
1305 (0X0519)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FCLIENT = %W, PRCL = %A, HWNDFRAME = %A

PMWIN Major Code: 0X00C2 Minor Code: 1306 (0X051A)

Description
WINDRAWBITMAP PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDRAWBITMAP

Minor Code
1306 (0X051A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
**Traced Parameters**

%FS = %W, CLRBACK = %D, CLRFORE = %D  
PPTLDST = %A, PWRCSRC = %A, HBM = %A, HPSDST = %A  
%TLDS->X = %D, PPTLDST->Y = %D  
%WRCSRC->XLEFT = %D, PWRCSRC->YBOTTOM = %D  
PWRCSRC->XRIGHT = %D, PWRCSRC->YTOP = %D

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 1307 (0X051B)**

**Description**  
WINDRAWBORDER PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINDRAWBORDER

**Minor Code**  
1307 (0X051B)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%RGFCMD = %W, CLRBACK = %D, CLRFORE = %D  
CY = %W, CX = %W, PRCL = %A, HPS = %A  
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D  
PRCL->XRIGHT = %D, PRCL->YTOP = %D

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 1308 (0X051C)**

**Description**  
WINGETMINPOSITION PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINGETMINPOSITION

**Minor Code**  
1308 (0X051C)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.
Traced Parameters

%TL = %A, PSWP = %A, HWND = %A
%TL->X = %D, PPTL->Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 1309 (0X051D)

Description
WINGETMAXPOSITION PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETMAXPOSITION

Minor Code
1309 (0X051D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%SWP = %A, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1350 (0X0546)

Description
WINFRAMEWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINFRAMEWNDPROC

Minor Code
1350 (0X0546)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1351 (0X0547)
WINSCROLLBARWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSCROLLBARWNDPROC

Minor Code
1351 (0X0547)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1352 (0X0548)

WINTITLEBARWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINTITLEBARWNDPROC

Minor Code
1352 (0X0548)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1353 (0X0549)

WINSTATICWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSTATICWNDPROC

Minor Code
1353 (0X0549)

Trace Groups
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1354 (0X054A)

Description
WINCONTEXTWNDPROC PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCONTEXTWNDPROC

Minor Code
1354 (0X054A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1355 (0X054B)

Description
WINCALLHELPHOOK PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCALLHELPHOOK

Minor Code
1355 (0X054B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RCPOSITION = %A, IDSUBTOPIC = %W
IDTOPIC = %W, SMODE = %W
%RCPOSITION->XLEFT = %D, PRCPOSITION->YBOTTOM = %D
PRCPOSITION->XRIGHT = %D, PRCPOSITION->YTOP = %D
### PMWIN Major Code: 0X00C2 Minor Code: 1356 (0X054C)

**Description**  
DESTROYLIST PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.DESTROYLIST

**Minor Code**  
1356 (0X054C)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

%LST = %W, HHEAP = %A

---

### PMWIN Major Code: 0X00C2 Minor Code: 1357 (0X054D)

**Description**  
INSERTLISTITEM PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.INSERTLISTITEM

**Minor Code**  
1357 (0X054D)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  

%CCHINSERT = %W, LPCHINSERT = %A, ILI = %W  
PLST = %W, HHEAP = %A

---

### PMWIN Major Code: 0X00C2 Minor Code: 1358 (0X054E)

**Description**  
DELETELISTITEM PRE-INVOCATION
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: PMWIN.DELETELISTITEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>1358 (0X054E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%ILI = %W, PLST = %W, HHEAP = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1359 (0X054F)

<table>
<thead>
<tr>
<th>Description</th>
<th>GETLISTITEMLENGTH PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.GETLISTITEMLENGTH</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1359 (0X054F)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%ILI = %W, PLST = %W, HHEAP = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1360 (0X0550)

<table>
<thead>
<tr>
<th>Description</th>
<th>GETLISTITEM PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.GETLISTITEM</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1360 (0X0550)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%ILI = %W, PLST = %W, HHEAP = %A</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 1362 (0X0552)

**Description**
SETLISTITEM PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.SETLISTITEM

**Minor Code**
1362 (0X0552)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%FRESIZE = %W, LPCGSRC = %A, CBSET = %W, IBLI = %W
ILI = %W, PLST = %W, HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 1400 (0X0578)

**Description**
WINLOADMENU POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADMENU

**Minor Code**
1400 (0X0578)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1401 (0X0579)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINCREatemenu Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WinCreateMenu</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1401 (0X0579)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

-------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1402 (0X057A)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINFlashWindow Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WinFlashWindow</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1402 (0X057A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

-------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 1403 (0X057B)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINCreateFrameControls Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEFrameControls</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1403 (0X057B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 1404 (0X057C)

Description
WINFORMATFRAME POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinFormatFrame

Minor Code
1404 (0X057C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1405 (0X057D)

Description
WINCALCFRAMERECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinCalcFrameRect

Minor Code
1405 (0X057D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1406 (0X057E)
Description: WINGETMINPOSITION POST-INVOCATION

Tracepoint:
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMINPOSITION

Minor Code:
1406 (0X057E)

Trace Groups:
No groups assigned.

Trace Types:
No types assigned.

Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1407 (0X057F)

Description: WINGETMAXPOSITION POST-INVOCATION

Tracepoint:
Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMAXPOSITION

Minor Code:
1407 (0X057F)

Trace Groups:
No groups assigned.

Trace Types:
No types assigned.

Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1500 (0X05DC)

Description: WINSETRECT PRE-INVOCATION

Tracepoint:
Public symbol defined dynamic tracepoint: PMWIN.WINSETRECT

Minor Code:
1500 (0X05DC)

Trace Groups:
No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%YTOP = %W, XRIGHT = %W, YBOTTOM = %W
XLEFT = %W, PRCL = %A
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1501 (0X05DD)

Description
WINISRECTEMPTY PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINISRECTEMPTY

Minor Code
1501 (0X05DD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%RCL = %A
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1502 (0X05DE)

Description
WINCOPYRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCOPYRECT

Minor Code
1502 (0X05DE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

%RCLSRC = %A, PRCLDST = %A
%RCLSRC->XLEFT = %D, PRCLSRC->YBOTTOM = %D
PRCLSRC->XRIGHT = %D, PRCLSRC->YTOP = %D
%RCLDST->XLEFT = %D, PRCLDST->YBOTTOM = %D
PRCLDST->XRIGHT = %D, PRCLDST->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1503 (0X05DF)

Description
WINEQUALRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINEQUALRECT

Minor Code
1503 (0X05DF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%RCL2 = %A, PRCL1 = %A
%RCL2->XLEFT = %D, PRCL2->YBOTTOM = %D
PRCL2->XRIGHT = %D, PRCL2->YTOP = %D
%RCL1->XLEFT = %D, PRCL1->YBOTTOM = %D
PRCL1->XRIGHT = %D, PRCL1->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1504 (0X05E0)

Description
WINSETRECTEMPTY PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETRECTEMPTY

Minor Code
1504 (0X05E0)

Trace Groups
No groups assigned.

Trace Types
PMWIN Major Code: 0X00C2 Minor Code: 1505 (0X05E1)

Description
WINOFFSETRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINOFFSETRECT

Minor Code
1505 (0X05E1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CY = %W, CX = %W, PRCL = %A
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1506 (0X05E2)

Description
WININFLATERECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WININFLATERECT

Minor Code
1506 (0X05E2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CY = %W, CX = %W, PRCL = %A
PMWIN Major Code: 0X00C2 Minor Code: 1507 (0X05E3)

Description
WINPTINRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINPTINRECT

Minor Code
1507 (0X05E3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%TL = %A, PRCL = %A
%TL->X = %D, PPTL->Y = %D
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1508 (0X05E4)

Description
WININTERSECTRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WININTERSECTRECT

Minor Code
1508 (0X05E4)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A
%RCLSRC2->XLEFT = %D, PRCLSRC1->YBOTTOM = %D
PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D
PMWIN Major Code: 0X00C2 Minor Code: 1509 (0X05E5)

Description
WINUNIONRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINUNIONRECT

Minor Code
1509 (0X05E5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A
%RCLSRC2->XLEFT = %D, PRCLSRC2->YBOTTOM = %D
PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D
%RCLSRC1->XLEFT = %D, PRCLSRC1->YBOTTOM = %D
PRCLSRC1->XRIGHT = %D, PRCLSRC1->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1510 (0X05E6)

Description
WINSUBTRACTRECT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSUBTRACTRECT

Minor Code
1510 (0X05E6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A
%RCLSRC2->XLEFT = %D, PRCLSRC2->YBOTTOM = %D
PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D
%RCLSRC1->XLEFT = %D, PRCLSRC1->YBOTTOM = %D
PMWIN Major Code: 0X00C2 Minor Code: 1511 (0X05E7)

**Description**
WININVERTRECT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WININVERTRECT

**Minor Code**
1511 (0X05E7)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%RCL = %A, HPS = %A
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D
PRCL->XRIGHT = %D, PRCL->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 1512 (0X05E8)

**Description**
WINMAKERECT PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINMAKERECT

**Minor Code**
1512 (0X05E8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%WRC = %A
%WRC->XLEFT = %W, PWRC->DUMMY1 = %W
PWRC->YBOTTOM = %W, PWRC->DUMMY2 = %W
PMWIN Major Code: 0X00C2 Minor Code: 1513 (0X05E9)

Description
WINMAKEPOINTS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINMAKEPOINTS

Minor Code
1513 (0X05E9)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CWPT = %W, PWPT = %A
%.WPT->X = %W, PWPT->DUMMY1 = %W
PWPT->Y = %W, PWPT->DUMMY2 = %W

PMWIN Major Code: 0X00C2 Minor Code: 1600 (0X0640)

Description
WINSETRECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETRECT

Minor Code
1600 (0X0640)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1601 (0X0641)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINISRECTEMPTY POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINISRECTEMPTY</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1601 (0X0641)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1602 (0X0642)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINCOPYRECT POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOPYRECT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1602 (0X0642)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1603 (0X0643)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINEQUALRECT POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINEQUALRECT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1603 (0X0643)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 1604 (0X0644)

Description
WINSETRECTEMPTY POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETRECTEMPTY

Minor Code
1604 (0X0644)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1605 (0X0645)

Description
WINOFFSETRECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINOFFSETRECT

Minor Code
1605 (0X0645)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1606 (0X0646)
**WININFLATERECT POST-INVOCATION**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWININFLATERECT

**Minor Code**
1606 (0X0646)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1607 (0X0647)

**Description**
WINPTINRECT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINPTINRECT

**Minor Code**
1607 (0X0647)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1608 (0X0648)

**Description**
WININTERSECTRECT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWININTERSECTRECT

**Minor Code**
1608 (0X0648)

**Trace Groups**
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 1609 (0X0649)

Description
WINUNIONRECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINUNIONRECT

Minor Code
1609 (0X0649)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1610 (0X064A)

Description
WINSUBTRACTRECT POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINUNSUBTRACTRECT

Minor Code
1610 (0X064A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1611 (0X064B)
**Description**
WINMAKERECT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAKERECT

**Minor Code**
1611 (0X064B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1612 (0X064C)

**Description**
WINMAKEPOINTS POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAKEPOINTS

**Minor Code**
1612 (0X064C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1613 (0X064D)

**Description**
WINTRACKRECT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinTrackRect

**Minor Code**
1613 (0X064D)

**Trace Groups**
No groups assigned.
**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1614 (0X064E)

**Description**
WINSHOWTRACKRECT POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWTRACKRECT

**Minor Code**
1614 (0X064E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1700 (0X06A4)

**Description**
WINQUERYVERSION PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYVERSION

**Minor Code**
1700 (0X06A4)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 1701 (0X06A5)
<table>
<thead>
<tr>
<th>Description</th>
<th>WININITIALIZE PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WININITIALIZE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1701 (0X06A5)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1702 (0X06A6)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINTERMINATE PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINTERMINATE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1702 (0X06A6)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%HAB = %A</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1703 (0X06A7)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYSYSTEMATOMTABLE PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSTEMATOMTABLE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1703 (0X06A7)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 1704 (0X06A8)

Description: WINQUERYSYSVALUE PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSVALUE
Minor Code: 1704 (0X06A8)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%ISYSVALUE = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 1705 (0X06A9)

Description: WINSETSYSVALUE PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSVALUE
Minor Code: 1705 (0X06A9)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%LVALUE = %D, ISYSVALUE = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 1706 (0X06AA)
Description: WINQUERYSYSCOLOR PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSCOLOR

Minor Code: 1706 (0X06AA)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%LRESERVED = %D, ICOLOR = %D, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 1707 (0X06AB)

Description: WINSETSYSCOLORS PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSCOLORS

Minor Code: 1707 (0X06AB)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%CLR = %A, CCLR = %D, CLRFIRST = %D
FLFORMAT = %D, FLOPTIONS = %D, HWNDDESKTOP = %A
%CLR -> %D

PMWIN Major Code: 0X00C2 Minor Code: 1708 (0X06AC)

Description: WINCATCH PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINCATCH

Minor Code:
1708 (0X06AC)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
\text{CATCHBUF} = \%A

--------------------------------------------

**PMWIN Major Code:** 0X00C2 **Minor Code:** 1709 (0X06AD)

**Description**
WINTHROW PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINTHROW

**Minor Code**
1709 (0X06AD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
\text{NTHROWBACK} = \%W, \text{PCATCHBUF} = \%A

\%
\text{CATCHBUF->CTCHBF} = \%W \%W \%W \%W \%W \%W \%W \%W

--------------------------------------------

**PMWIN Major Code:** 0X00C2 **Minor Code:** 1710 (0X06AE)

**Description**
WINGETLASTERROR PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINGETLASTERROR

**Minor Code**
1710 (0X06AE)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
\text{HAB} = \%A
PMWIN Major Code: 0X00C2 Minor Code: 1711 (0X06AF)

**Description**
WINGETERRORINFO PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINGETERRORINFO

**Minor Code**
1711 (0X06AF)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 1712 (0X06B0)

**Description**
WINFREEERRORINFO PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINFREEERRORINFO

**Minor Code**
1712 (0X06B0)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%ERRINFO = %A
%ERRINFO->CBFIXEDERRINFO = %W
PERRINFO->IDERROR = %D, PERRINFO->CDETAILLEVEL = %W
PERRINFO->OFFAOFFSZMSG = %W, PERRINFO->OFFBINARYDATA = %W

PMWIN Major Code: 0X00C2 Minor Code: 1713 (0X06B1)
WINSTARTTIMER PRE-INVOCATION

Description
WINSTARTTIMER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSTARTTIMER

Minor Code
1713 (0X06B1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%DTTIMEOUT = %W, IDTIMER = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1714 (0X06B2)

WINSTOPTIMER PRE-INVOCATION

Description
WINSTOPTIMER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSTOPTIMER

Minor Code
1714 (0X06B2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%IDTIMER = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 1715 (0X06B3)

WINGETCURRENTTIME PRE-INVOCATION

Description
WINGETCURRENTTIME PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINGETCURRENTTIME

Minor Code
1715 (0X06B3)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 1750 (0X06D6)

Description
WINSETERRORINFO PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN._WINSETERRORINFO

Minor Code
1750 (0X06D6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RETADDR = %A, ERRCODE = %D, FOPTIONS = %W, ARGS = %W

PMWIN Major Code: 0X00C2 Minor Code: 1751 (0X06D7)

Description
WINSETLASTERROR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETLASTERROR

Minor Code
1751 (0X06D7)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%RETADDR = %A, ERRCODE = %D

PMWIN Major Code: 0X00C2 Minor Code: 1800 (0X0708)
Description
WINQUERYSYSVALUE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSVALUE

Minor Code
1800 (0X0708)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1801 (0X0709)

Description
WINSETSYSVALUE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETSYSVALUE

Minor Code
1801 (0X0709)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1802 (0X070A)

Description
WINQUERYSYSCOLOR POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinQuerySysColor

Minor Code
1802 (0X070A)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 1803 (0X070B)

Description
WINSETSYSCOLORS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetSysColors

Minor Code
1803 (0X070B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1804 (0X070C)

Description
WINCATCH POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINCATCH

Minor Code
1804 (0X070C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1805 (0X070D)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINTHROW POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINTHROW</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1805 (0X070D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1806 (0X070E)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETLASTERROR POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WinGetLastError</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1806 (0X070E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1807 (0X070F)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETERRORINFO POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WinGetErrorInfo</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1807 (0X070F)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 1808 (0X0710)

Description
WINFREEERRORINFO POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinFreeErrorInfo

Minor Code
1808 (0X0710)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1809 (0X0711)

Description
WINSTARTTIMER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSTARTTIMER

Minor Code
1809 (0X0711)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 1810 (0X0712)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINSTOPTIMER POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINSTOPTIMER</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1810 (0X0712)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1811 (0X0713)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINGETCURRENTTIME POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETCURRENTTIME</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1811 (0X0713)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 1900 (0X076C)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINLOADACCELTABLE PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINLOADACCELTABLE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1900 (0X076C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
Trace Types

No types assigned.

Traced Parameters

%IDACCELTABLE = %W, HMOD = %W

PMWIN Major Code: 0X00C2 Minor Code: 1901 (0X076D)

Description

WINCREATEACCELTABLE PRE-INVOCATION

Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINCREATEACCELTABLE

Minor Code

1901 (0X076D)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%ACCELTABLE = %A

%ACCELTABLE->CACCEL = %W, PACCELTABLE->CODEPAGE = %W

PACCELTABLE->FS = %W, PACCELTABLE->KEY = %W

PACCELTABLE->CMD= %W

PMWIN Major Code: 0X00C2 Minor Code: 1902 (0X076E)

Description

WINDESTROYACCELTABLE PRE-INVOCATION

Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYACCELTABLE

Minor Code

1902 (0X076E)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

%HACCEL = %A
PMWIN Major Code: 0X00C2 Minor Code: 1903 (0X076F)

Description
WINCOPYACCELTABLE PRE-INVOCAUTION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCOPYACCELTABLE

Minor Code
1903 (0X076F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CBCOPYMAX = %W, PACCELTABLE = %A, HACCEL = %A

PMWIN Major Code: 0X00C2 Minor Code: 1904 (0X0770)

Description
WINTRANSLATEACCEL PRE-INVOCAUTION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINTRANSLATEACCEL

Minor Code
1904 (0X0770)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%QMSG = %A, HACCEL = %A, HWND = %A
%QMSG->HWND = %A, PQMSG->MSG = %W, PQMSG->MP1 = %D
PQMSG->MP2 = %D, PQMSG->TIME = %D, PQMSG->PTL.X = %D
PQMSG->PTL.Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 1905 (0X0771)
WINSETACCELTABLE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETACCELTABLE

Minor Code
1905 (0X0771)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDFRAME = %A, HACCEL = %A

PMWIN Major Code: 0X00C2 Minor Code: 1906 (0X0772)

WINQUERYACCELTABLE PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYACCELTABLE

Minor Code
1906 (0X0772)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDFRAME = %A

PMWIN Major Code: 0X00C2 Minor Code: 2000 (0X07D0)

WINLOADACCELTABLE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADACCELTABLE

Minor Code
2000 (0X07D0)

Trace Groups
No groups assigned.
Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, \ AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2001 (0X07D1)

Description: WINCREATEACCELTABLE POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEACCELTABLE

Minor Code: 2001 (0X07D1)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, \ AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2002 (0X07D2)

Description: WINDESTROYACCELTABLE POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYACCELTABLE

Minor Code: 2002 (0X07D2)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, \ AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2003 (0X07D3)
**WINCOPYACCELTABLE POST-INVOCATION**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOPYACCELTABLE

**Minor Code**
2003 (0X07D3)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2004 (0X07D4)

**WINTRANSLATEACCEL POST-INVOCATION**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINTRANSLATEACCEL

**Minor Code**
2004 (0X07D4)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2005 (0X07D5)

**WINSETACCELTABLE POST-INVOCATION**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETACCELTABLE

**Minor Code**
2005 (0X07D5)

**Trace Groups**
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2006 (0X07D6)

Description
WINQUERYACCELTABLE POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYACCELTABLE

Minor Code
2006 (0X07D6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2100 (0X0834)

Description
WINOPENCLIPBRD PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINOPENCLIPBRD

Minor Code
2100 (0X0834)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 2101 (0X0835)
WINCLOSECLIPBRD PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINCLOSECLIPBRD

Minor Code
2101 (0X0835)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 2102 (0X0836)

WINEMPTYCLIPBRD PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINEMPTYCLIPBRD

Minor Code
2102 (0X0836)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HAB = %A

PMWIN Major Code: 0X00C2 Minor Code: 2103 (0X0837)

WINSETCLIPBRDOWNER PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDOWNER

Minor Code
2103 (0X0837)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2104 (0X0838)

**Description**
WINQUERYCLIPBRDOWNER PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDOWNER

**Minor Code**
2104 (0X0838)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 2105 (0X0839)

**Description**
WINSETCLIPBRDDATA PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDDATA

**Minor Code**
2105 (0X0839)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%FLOCK = %W

PMWIN Major Code: 0X00C2 Minor Code: 2106 (0X083A)

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDDATA

**Minor Code**
2106 (0X083A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%RGFFMTINFO = %W, FMT = %W, ULDATA = %D
<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYCLIPBRDDATA PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDDATA</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2106 (0X083A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%FMT = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 2107 (0X083B)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINENUMCLIPBRDFMTS PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINENUMCLIPBRDFMTS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2107 (0X083B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%FMT = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 2108 (0X083C)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINQUERYCLIPBRDFMTINFO PRE-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDFMTINFO</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2108 (0X083C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 2109 (0X083D)

Description
WINSETCLIPBRDVIEWER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDVIEWER

Minor Code
2109 (0X083D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HWNDNEWCLIPVIEWER = %A

PMWIN Major Code: 0X00C2 Minor Code: 2110 (0X083E)

Description
WINQUERYCLIPBRDVIEWER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDVIEWER

Minor Code
2110 (0X083E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FLOCK = %W
PMWIN Major Code: 0X00C2 Minor Code: 2200 (0X0898)

Description          WINOPENCLIPBRD POST-INVOCATION
Tracepoint           Public symbol defined dynamic tracepoint: PMWIN.WinOpenClipbrd
Minor Code           2200 (0X0898)
Trace Groups         No groups assigned.
Trace Types          No types assigned.
Traced Parameters    DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2201 (0X0899)

Description          WINCLOSECLIPBRD POST-INVOCATION
Tracepoint           Public symbol defined dynamic tracepoint: PMWIN.WinCloseClipbrd
Minor Code           2201 (0X0899)
Trace Groups         No groups assigned.
Trace Types          No types assigned.
Traced Parameters    DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2202 (0X089A)

Description          WINEMPTYCLIPBRD POST-INVOCATION
Tracepoint           Public symbol defined dynamic tracepoint: PMWIN.WinEmptyClipbrd
Minor Code           2202 (0X089A)
PMWIN Major Code: 0X00C2 Minor Code: 2203 (0X089B)

Description
WINSETCLIPBRDOWNER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdOwner

Minor Code
2203 (0X089B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

-----------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2204 (0X089C)

Description
WINQUERYCLIPBRDOWNER/WINQUERYCLIPBRDVIEWER/WINQUERYFOCUS/WINQUERYSYSMODALWINDOW POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLIPBRDOWNER

Minor Code
2204 (0X089C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 2205 (0X089D)

Description
WINSETCLIPBRDDATA POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdData

Minor Code
2205 (0X089D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2206 (0X089E)

Description
WINQUERYCLIPBRDDATA POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinQueryClipbrdData

Minor Code
2206 (0X089E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2207 (0X089F)

Description
WINENUMCLIPBRDFMTS POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinEnumClipbrdFmts
PMWIN Major Code: 0X00C2 Minor Code: 2208 (0X08A0)

Description
WINQUERYCLIPBRDFMTINFO POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinQueryClipbrdFmtInfo

Minor Code
2208 (0X08A0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2209 (0X08A1)

Description
WINSETCLIPBRDVIEWER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdViewer

Minor Code
2209 (0X08A1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W
PMWIN Major Code: 0X00C2 Minor Code: 2210 (0X08A2)

Description
WINQUERYCLIPBRDVIEWER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLIPBRDVIEWER

Minor Code
2210 (0X08A2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2300 (0X08FC)

Description
WINDESTROYCURSOR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYCURSOR

Minor Code
2300 (0X08FC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 2301 (0X08FD)

Description
WINSHOWCURSOR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSHOWCURSOR
PMWIN Major Code: 0X00C2 Minor Code: 2301 (0X08FD)

**Minor Code**
2301 (0X08FD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%FSHOW = %W, HWND = %A

PMWIN Major Code: 0X00C2 Minor Code: 2302 (0X08FE)

**Description**
WINCREATECURSOR PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINCREATECURSOR

**Minor Code**
2302 (0X08FE)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%RCLCLIP = %A, FS = %W, CY = %W
CX = %W, Y = %W, X = %W, HWND = %A
%RCLCLIP->XLEFT = %D, PRCLCLIP->YBOTTOM = %D
PRCLCLIP->XRIGHT = %D, PRCLCLIP->YTOP = %D

PMWIN Major Code: 0X00C2 Minor Code: 2303 (0X08FF)

**Description**
WINQUERYCURSORINFO PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCURSORINFO

**Minor Code**
2303 (0X08FF)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

%CURSORINFO = %A, HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2304 (0X0900)

Description
WINSETPOINTER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETPOINTER

Minor Code
2304 (0X0900)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HPTRNEW = %A, HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2305 (0X0901)

Description
WINSHOWPOINTER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSHOWPOINTER

Minor Code
2305 (0X0901)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%FSHOW = %W, HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2306 (0X0902)
**Description**  
WINQUERYSYSPOINTER PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSPOINTER

**Minor Code**  
2306 (0X0902)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

$%FLOAD = %W, IPTR = %W, HWNDDESKTOP = %A$

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 2307 (0X0903)**

**Description**  
WINLOADPOINTER PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINLOADPOINTER

**Minor Code**  
2307 (0X0903)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

$%IDRES = %W, HMOD = %W, HWNDDESKTOP = %A$

--------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 2308 (0X0904)**

**Description**  
WINCREATEPOINTER PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINCREATEPOINTER

**Minor Code**  
2308 (0X0904)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.
Traced Parameters

%YHOTSPOT = %W, XHOTSPOT = %W, FPOINTER = %W
HBMPOINTER = %A, HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2309 (0X0905)

Description
WINDESTROYPOINTER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYPOINTER

Minor Code
2309 (0X0905)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HPT = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2310 (0X0906)

Description
WINQUERYPOINTER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTER

Minor Code
2310 (0X0906)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HWNDDESKTOP = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2311 (0X0907)
WINSETPOINTERPOS PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINSETPOINTERPOS

Minor Code
2311 (0x0907)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%Y = \%W, X = \%W, HWNDDESKTOP = \%A

PMWIN Major Code: 0x00C2 Minor Code: 2312 (0x0908)

WINQUERYPOINTERPOS PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTERPOS

Minor Code
2312 (0x0908)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%TL = \%A, HWNDDESKTOP = \%A

PMWIN Major Code: 0x00C2 Minor Code: 2313 (0x0909)

WINQUERYPOINTERINFO PRE-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTERINFO

Minor Code
2313 (0x0909)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2314 (0X090A)

Description: WINDRAWPOINTER PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINDRAWPOINTER
Minor Code: 2314 (0X090A)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: %FS = %W, HPTR = %A, Y = %W, X = %W, HPS = %A

PMWIN Major Code: 0X00C2 Minor Code: 2315 (0X090B)

Description: WINGETSYSBITMAP PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINGETSYSBITMAP
Minor Code: 2315 (0X090B)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: %IBM = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2400 (0X0960)
<table>
<thead>
<tr>
<th>Description</th>
<th>WINDESTROYCURSOR POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYCURSOR</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2400 (0X0960)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 2401 (0X0961)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINSHOWCURSOR POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWCURSOR</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2401 (0X0961)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>DX = %W, AX = %W</td>
</tr>
</tbody>
</table>

PMWIN Major Code: 0X00C2 Minor Code: 2402 (0X0962)

<table>
<thead>
<tr>
<th>Description</th>
<th>WINCREATECURSOR POST-INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATECURSOR</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2402 (0X0962)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMWIN Major Code: 0X00C2 Minor Code: 2403 (0X0963)

Description
WINQUERYCURSORINFO POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCURSORINFO

Minor Code
2403 (0X0963)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2404 (0X0964)

Description
WINSETPOINTER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPOINTER

Minor Code
2404 (0X0964)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2405 (0X0965)
WINSHOWPOINTER POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWPOINTER

Minor Code

2405 (0X0965)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2406 (0X0966)

WINQUERYSYSPINTER POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSPINTER

Minor Code

2406 (0X0966)

Trace Groups

No groups assigned.

Trace Types

No types assigned.

Traced Parameters

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2407 (0X0967)

WINLOADPOINTER POST-INVOCATION

Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADPOINTER

Minor Code

2407 (0X0967)

Trace Groups

No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2408 (0X0968)

Description: WINCREATEPOINTER POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEPOINTER
Minor Code: 2408 (0X0968)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2409 (0X0969)

Description: WINDESTROYPOINTER POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYPOINTER
Minor Code: 2409 (0X0969)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2410 (0X096A)
Description: WINQUERYPOINTER POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTER

Minor Code: 2410 (0X096A)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2411 (0X096B)

Description: WINSETPOINTERPOS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPOINTERPOS

Minor Code: 2411 (0X096B)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2412 (0X096C)

Description: WINQUERYPOINTERPOS POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTERPOS

Minor Code: 2412 (0X096C)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2413 (0X096D)

Description
WINQUERYPOINTERINFO POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTERINFO

Minor Code
2413 (0X096D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2414 (0X096E)

Description
WINDRAWPOINTER POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.POSTWINDRAWPOINTER

Minor Code
2414 (0X096E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2415 (0X096F)
Description: WINGETSYSBITMAP POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETSYSBITMAP

Minor Code: 2415 (0X096F)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters: 

DX = %W, AX = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2500 (0X09C4)

Description: WINSETHOOK PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINSETHOOK

Minor Code: 2500 (0X09C4)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters: 

%HMOD = %W, PFNHOOK = %A, IHOOK = %W, HMQ = %A

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2501 (0X09C5)

Description: WINRELEASEHOOK PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINRELEASEHOOK

Minor Code: 2501 (0X09C5)

Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2502 (0X09C6)

**Description**
WINCALLMSGFILTER PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINCALLMSGFILTER

**Minor Code**
2502 (0X09C6)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%MSGF = %W, PQMSG = %A

%QMSG->HWND = %A, PQMSG->MSG = %W

PQMSG->MP1 = %D, PQMSG->MP2 = %D

PQMSG->TIME = %D, PQMSG->PTL.X = %D

PQMSG->PTL.Y = %D

PMWIN Major Code: 0X00C2 Minor Code: 2552 (0X09F8)

**Description**
FARCALLHOOK PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.FARCALLHOOK

**Minor Code**
2552 (0X09F8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
PMWIN Major Code: 0X00C2 Minor Code: 2553 (0X09F9)

Description
FREEQUEUEWINDOWHOOKS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.FREEQUEUEWINDOWHOOKS

Minor Code
2553 (0X09F9)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%FBADEXIT = %W, SMQ = %W

PMWIN Major Code: 0X00C2 Minor Code: 2600 (0X0A28)

Description
WINSETHOOK POST-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WinSetHook

Minor Code
2600 (0X0A28)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2601 (0X0A29)

Description
WINRELEASEHOOK POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinReleaseHook

Minor Code: 2601 (0X0A29)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2602 (0X0A2A)

Description: WINCALLMSGFILTER POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinCallMsgFilter

Minor Code: 2602 (0X0A2A)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\[ DX = \%W, AX = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2700 (0X0A8C)

Description: WINSETCP PRE-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinSetTCP

Minor Code: 2700 (0X0A8C)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:
PMWIN Major Code: 0X00C2 Minor Code: 2701 (0X0A8D)

Description
WINQUERYCP PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCP

Minor Code
2701 (0X0A8D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 2702 (0X0A8E)

Description
WINQUERYCPLIST PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCPLIST

Minor Code
2702 (0X0A8E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%RGCP = %A, CCPMAX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2703 (0X0A8F)

Description
WINCPTRANSLATESTRING PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCPTRANSLATESTRING

Minor Code
2703 (0X0A8F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CHDEST = %A, CCHDESTMAX = %W
CPDST = %W, PSZSRC = %A, CPSRC = %W
PSZSRC -> %S

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2704 (0X0A90)

Description
WINCPTRANSFORMCHAR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCPTRANSFORMCHAR

Minor Code
2704 (0X0A90)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%CPDST = %W, CHSRC = %B, CPSRC = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2705 (0X0A91)

Description
WINUPPER PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINUPPER

Minor Code
2705 (0X0A91)

Trace Groups
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2706 (0X0A92)

Description
WINUPPERCHAR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINUPPERCHAR

Minor Code
2706 (0X0A92)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%SZ = %A, IDCC = %W, IDCP = %W
PSZ -> %S

PMWIN Major Code: 0X00C2 Minor Code: 2707 (0X0A93)

Description
WINNEXTCHAR PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINNEXTCHAR

Minor Code
2707 (0X0A93)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%SZ = %A, IDCC = %W, IDCP = %W
PSZ -> %S
PMWIN Major Code: 0X00C2 Minor Code: 2708 (0X0A94)

**Description**
WINPREVCHAR PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINPREVCHAR

**Minor Code**
2708 (0X0A94)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%SZ = %A, PSZSTART = %A, IDCC = %W, IDCP = %W
PSZ -> %S
PSZSTART -> %S

PMWIN Major Code: 0X00C2 Minor Code: 2709 (0X0A95)

**Description**
WINCOMPARESTRINGS PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINCOMPARESTRINGS

**Minor Code**
2709 (0X0A95)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%RESERVED = %W, PSZ2 = %A, PSZ1 = %A
IDCC = %W, IDCP = %W
PSZ2 -> %S
PSZ1 -> %S

PMWIN Major Code: 0X00C2 Minor Code: 2710 (0X0A96)
Description: WINLOADSTRING PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINLOADSTRING
Minor Code: 2710 (0X0A96)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%CHBUFFER = %A, CCHMAX = %W, ID = %W, HMOD = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2711 (0X0A97)

Description: WINLOADMESSAGE PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINLOADMESSAGE
Minor Code: 2711 (0X0A97)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%CHBUFFER = %A, CCHMAX = %W, ID = %W, HMOD = %W

--------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2750 (0X0ABE)

Description: WINLOADCHARXLATETBL PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WINLOADCHARXLATETBL
Minor Code: 2750 (0X0ABE)
Trace Groups: No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2751 (0X0ABF)

Description
WINSETCHARLATETBL PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETCHARLATETBL

Minor Code
2751 (0X0ABF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%LPXLATETBL = %A, HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 2752 (0X0AC0)

Description
WINQUERYCHARLATETBL PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCHARLATETBL

Minor Code
2752 (0X0AC0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HMQ = %A

PMWIN Major Code: 0X00C2 Minor Code: 2753 (0X0AC1)
Description  WINLOADVKEYGLYPHXLATETBL PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINLOADVKEYGLYPHXLATETBL
Minor Code  2753 (0X0AC1)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  
%IDOUTPUT = %W, IDLANGUAGE = %W, IDKBDTYPE = %W, HMOD = %W

PMWIN Major Code: 0X00C2 Minor Code: 2754 (0X0AC2)

Description  WINSETVKEYGLYPHXLATETBL PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINSETVKEYGLYPHXLATETBL
Minor Code  2754 (0X0AC2)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters  
%IDOUTPUT = %W, LPXLATETBL = %A

PMWIN Major Code: 0X00C2 Minor Code: 2755 (0X0AC3)

Description  WINQUERYVKEYGLYPHXLATETBL PRE-INVOCATION
Tracepoint  Public symbol defined dynamic tracepoint: PMWIN.WINQUERYVKEYGLYPHXLATETBL
Minor Code  2755 (0X0AC3)
Trace Groups  No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2756 (0X0AC4)

Description
WINSETKBDLAYOUT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINSETKBDLAYOUT

Minor Code
2756 (0X0AC4)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%IDKBDLAYOUT = %W, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2757 (0X0AC5)

Description
WINQUERYKBDLAYOUT PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYKBDLAYOUT

Minor Code
2757 (0X0AC5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2800 (0X0AF0)
**Description**
WINSETCP POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinSetCp

**Minor Code**
2800 (0X0AF0)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2801 (0X0AF1)

---

**Description**
WINQUERYCP POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinQueryCp

**Minor Code**
2801 (0X0AF1)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2802 (0X0AF2)

---

**Description**
WINQUERYCPLIST POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinQueryCpList

**Minor Code**
2802 (0X0AF2)

**Trace Groups**
No groups assigned.
PMWIN Major Code: 0X00C2 Minor Code: 2803 (0X0AF3)

Description: WINCPRTRANSLATESTRING POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinCpTranslateString
Minor Code: 2803 (0X0AF3)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2804 (0X0AF4)

Description: WINCPRTRANSLATECHAR POST-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinCpTranslateChar
Minor Code: 2804 (0X0AF4)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters: DX = %W, AX = %W

PMWIN Major Code: 0X00C2 Minor Code: 2805 (0X0AF5)
Description: WINUPPER POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinUpper

Minor Code: 2805 (0X0AF5)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

---------------------------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2806 (0X0AF6)

Description: WINUPPERCHAR POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinUpperChar

Minor Code: 2806 (0X0AF6)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

DX = %W, AX = %W

---------------------------------------------------------------

PMWIN Major Code: 0X00C2 Minor Code: 2807 (0X0AF7)

Description: WINNEXTCHAR POST-INVOCATION

Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.WinNextChar

Minor Code: 2807 (0X0AF7)

Trace Groups: No groups assigned.
**PMWIN Major Code: 0X00C2 Minor Code: 2808 (0X0AF8)**

**Description**
WINPREVCHAR POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WinPrevChar

**Minor Code**
2808 (0X0AF8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = \%W, AX = \%W \]

--------------------------------------------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 2809 (0X0AF9)**

**Description**
WINCOMPARESTRINGS POST-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOMPARESTRINGS

**Minor Code**
2809 (0X0AF9)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DX = \%W, AX = \%W \]

--------------------------------------------------------------------------------

**PMWIN Major Code: 0X00C2 Minor Code: 2900 (0X0B54)**
WINCREATEHEAP PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINCREATEHEAP

Minor Code
2900 (0X0B54)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

- %FOPTIONS = %W, CBMAXDED = %W, CHMINDED = %W
- CBGROW = %W, CBHEAP = %W, SELHEAPBASE = %W

PMWIN Major Code: 0X00C2 Minor Code: 2901 (0X0B55)

WINDESTROYHEAP PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYHEAP

Minor Code
2901 (0X0B55)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

- %HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2902 (0X0B56)

WINAVAILMEM PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINAVAILMEM

Minor Code
2902 (0X0B56)
Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CBMINFREE = %W, FCOMPACT = %W, HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2903 (0X0B57)

Description
WINALLOCMEM PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINALLOCMEM

Minor Code
2903 (0X0B57)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CB = %W, HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2904 (0X0B58)

Description
WINREALLOCMEM PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.WINREALLOCMEM

Minor Code
2904 (0X0B58)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%CBNEW = %W, CBOLD = %W, NPMEM = %W, HHEAP = %A
PMWIN Major Code: 0X00C2 Minor Code: 2905 (0X0B59)

**Description**
WINFREEMEM PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINFREEMEM

**Minor Code**
2905 (0X0B59)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%CBMEM = %W, NPMEM = %W, HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2906 (0X0B5A)

**Description**
WINLOCKHEAP PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.WINLOCKHEAP

**Minor Code**
2906 (0X0B5A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%HHEAP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2950 (0X0B86)

**Description**
COMPACTMOVEABLEHEAP PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.COMPACTMOVEABLEHEAP

**Minor Code**
2950 (0X0B86)
PMWIN Major Code: 0X00C2 Minor Code: 2951 (0X0B87)

Description: FINDFREEBLOCK PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.FINDFREEBLOCK
Minor Code: 2951 (0X0B87)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

PHEAPCB = %A, CBLOCKSIZE = %W

PMWIN Major Code: 0X00C2 Minor Code: 2952 (0X0B88)

Description: FINDMAXFREEBLOCK PRE-INVOCATION
Tracepoint: Public symbol defined dynamic tracepoint: PMWIN.FINDMAXFREEBLOCK
Minor Code: 2952 (0X0B88)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

PHEAPCB = %A
PMWIN Major Code: 0X00C2 Minor Code: 2953 (0X0B89)

Description
INSERTFREEBLOCK PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.INSERTFREEBLOCK

Minor Code
2953 (0X0B89)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PHEAPCB = %A, PBLOCK = %W, CBBLOCKSIZE = %W

PMWIN Major Code: 0X00C2 Minor Code: 2954 (0X0B8A)

Description
SORTFREELIST PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.SORTFREELIST

Minor Code
2954 (0X0B8A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PHEAPCB = %A, PFREELISTHEAD = %A, CBBLOCKSIZE = %W

PMWIN Major Code: 0X00C2 Minor Code: 2955 (0X0B8B)

Description
GETSIZEDS PRE-INVOCATION

Tracepoint
Public symbol defined dynamic tracepoint: PMWIN.GETSIZEDS

Minor Code
2955 (0X0B8B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[ DS = \%W \]

PMWIN Major Code: 0X00C2 Minor Code: 2956 (0X0B8C)

**Description**
VALIDATEHEAPHANDLE PRE-INVOCATION

**Tracepoint**
Public symbol defined dynamic tracepoint: PMWIN.VALIDATEHEAPHANDLE

**Minor Code**
2956 (0X0B8C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
No parameters traced.

PMGRE.DLL Trace Events

The tracepoints for the PMGRE.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

**Delay:**
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMGRE Major Code: 0X00C3, sorted by minor code.
Trace events for PMGRE Major Code: 0X00C3, sorted by tracepoint.

Trace Events for PMGRE Major Code: 0X00C3, Sorted by Minor Code

00002 (0X0002) GREENTRY2 Pre-Invocation
00003 (0X0003) GREENTRY3 Pre-Invocation
00004 (0X0004) GREENTRY4 Pre-Invocation
Trace Events for PMGRE Major Code: 0X00C3, Sorted by Tracepoint

PMGRE Major Code: 0X00C3 Minor Code: 2 (0X0002)
Description: GREENTRY2 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GREENTRY2

Minor Code: 2 (0X0002)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %a, lparam2 = %d, lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 3 (0X0003)

Description: GREENTRY3 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GREENTRY3

Minor Code: 3 (0X0003)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %a, lparam3 = %d, lparam2 = %d

lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 4 (0X0004)

Description: GREENTRY4 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GREENTRY4

Minor Code: 4 (0X0004)

Trace Groups: No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%Return Address = %a, lparam4 = %d, lparam3 = %d
lparam2 = %d, lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 5 (0X0005)

Description
GREENTRY5 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY5

Minor Code
5 (0X0005)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %a, lparam5 = %d, lparam4 = %d
lparam3 = %d, lparam2 = %d, lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 6 (0X0006)

Description
GREENTRY6 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY6

Minor Code
6 (0X0006)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %a, lparam6 = %d, lparam5 = %d
lparam4 = %d, lparam3 = %d, lparam2 = %d
PMGRE Major Code: 0X00C3 Minor Code: 7 (0X0007)

Description
GREENTRY7 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY7

Minor Code
7 (0X0007)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %a, lparam7 = %d, lparam6 = %d
lparam5 = %d, lparam4 = %d, lparam3 = %d,
lparam2 = %d, lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 8 (0X0008)

Description
GREENTRY8 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY8

Minor Code
8 (0X0008)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %a, lparam8 = %d, lparam7 = %d
lparam6 = %d, lparam5 = %d, lparam4 = %d,
lparam3 = %d, lparam2 = %d, lparam1 = %d

PMGRE Major Code: 0X00C3 Minor Code: 9 (0X0009)
**Description**
GREENTRY9 Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY9

**Minor Code**
9 (0X0009)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\begin{align*}
\%\text{Return Address} & = \%a, \, \text{lparam9} = \%d, \text{lparam8} = \%d \\
\text{lparam7} & = \%d, \, \text{lparam6} = \%d, \text{lparam5} = \%d \\
\text{lparam4} & = \%d, \, \text{lparam3} = \%d, \text{lparam2} = \%d \\
\text{lparam1} & = \%d
\end{align*}
\]

--------------------------------------------

PMGRE Major Code: 0X00C3 Minor Code: 10 (0X000A)

**Description**
GREENTRY10 Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGRE.GREENTRY10

**Minor Code**
10 (0X000A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\[
\begin{align*}
\%\text{Return Address} & = \%a, \, \text{lparam10} = \%d, \text{lparam9} = \%d \\
\text{lparam8} & = \%d, \, \text{lparam7} = \%d, \text{lparam6} = \%d, \\
\text{lparam5} & = \%d, \, \text{lparam4} = \%d, \text{lparam3} = \%d, \\
\text{lparam2} & = \%d, \, \text{lparam1} = \%d
\end{align*}
\]

--------------------------------------------

PMGRE Major Code: 0X00C3 Minor Code: 12 (0X000C)
Description: GRE32ENTRY2 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY2

Minor Code: 12 (0X000C)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d

PMGRE Major Code: 0X00C3 Minor Code: 13 (0X000D)

Description: GRE32ENTRY3 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY3

Minor Code: 13 (0X000D)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d

PMGRE Major Code: 0X00C3 Minor Code: 14 (0X000E)

Description: GRE32ENTRY4 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY4

Minor Code: 14 (0X000E)

Trace Groups: No groups assigned.
Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d

PMGRE Major Code: 0X00C3 Minor Code: 15 (0X000F)

Description: GRE32ENTRY5 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY5

Minor Code: 15 (0X000F)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d

PMGRE Major Code: 0X00C3 Minor Code: 16 (0X0010)

Description: GRE32ENTRY6 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY6

Minor Code: 16 (0X0010)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
PMGRE Major Code: 0X00C3 Minor Code: 17 (0X0011)

Description
GRE32ENTRY7 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY7

Minor Code
17 (0X0011)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d

PMGRE Major Code: 0X00C3 Minor Code: 18 (0X0012)

Description
GRE32ENTRY8 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY8

Minor Code
18 (0X0012)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d

PMGRE Major Code: 0X00C3 Minor Code: 19 (0X0013)
Description: GRE32ENTRY9 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY9

Minor Code: 19 (0X0013)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d
lparam9 = %d

PMGRE Major Code: 0X00C3 Minor Code: 20 (0X0014)

Description: GRE32ENTRY10 Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY10

Minor Code: 20 (0X0014)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d
lparam9 = %d, lparam10 = %d

PMGRE Major Code: 0X00C3 Minor Code: 22 (0X0016)
INNERGRE32ENTRY2 Pre-Invocation

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY2

Minor Code
22 (0X0016)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d

PMGRE Major Code: 0X00C3 Minor Code: 23 (0X0017)

INNERGRE32ENTRY3 Pre-Invocation

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY3

Minor Code
23 (0X0017)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d

PMGRE Major Code: 0X00C3 Minor Code: 24 (0X0018)

INNERGRE32ENTRY4 Pre-Invocation

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY4

Minor Code
24 (0X0018)

Trace Groups
No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, Ipam1 = %d, Ipam2 = %d
Ipam3 = %d, Ipam4 = %d

PMGRE Major Code: 0X00C3 Minor Code: 25 (0X0019)

Description
INNERGRE32ENTRY5 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY5

Minor Code
25 (0X0019)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, Ipam1 = %d, Ipam2 = %d
Ipam3 = %d, Ipam4 = %d, Ipam5 = %d

PMGRE Major Code: 0X00C3 Minor Code: 26 (0X001A)

Description
INNERGRE32ENTRY6 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY6

Minor Code
26 (0X001A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, Ipam1 = %d, Ipam2 = %d
Ipam3 = %d, Ipam4 = %d, Ipam5 = %d
PMGRE Major Code: 0X00C3 Minor Code: 27 (0X001B)

Description
INNERGRE32ENTRY7 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY7

Minor Code
27 (0X001B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d

PMGRE Major Code: 0X00C3 Minor Code: 28 (0X001C)

Description
INNERGRE32ENTRY8 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY8

Minor Code
28 (0X001C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d

PMGRE Major Code: 0X00C3 Minor Code: 29 (0X001D)
INNERGRE32ENTRY9 Pre-Invocation

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY9

Minor Code
29 (0X001D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d
lparam9 = %d

PMGRE Major Code: 0X00C3 Minor Code: 30 (0X001E)

INNERGRE32ENTRY10 Pre-Invocation

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY10

Minor Code
30 (0X001E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Return Address = %f, lparam1 = %d, lparam2 = %d
lparam3 = %d, lparam4 = %d, lparam5 = %d
lparam6 = %d, lparam7 = %d, lparam8 = %d
lparam9 = %d

PMPIC.DLL Trace Events

The tracepoints for the PMPIC.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.
Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMPIC Major Code: 0X00C4, sorted by minor code.
Trace events for PMPIC Major Code: 0X00C4, sorted by tracepoint.

Trace Events for PMPIC Major Code: 0X00C4, Sorted by Minor Code

00010 (0X000A) PicIchg Pre-Invocation
01000 (0X03E8) PicPrint Pre-Invocation

Trace Events for PMPIC Major Code: 0X00C4, Sorted by Tracepoint

PICICHG 00010 (0X000A)
PICPRINT 01000 (0X03E8)

PMPIC Major Code: 0X00C4 Minor Code: 10 (0X000A)

Description      PicIchg Pre-Invocation
Tracepoint       Public symbol defined dynamic tracepoint: PMPIC.PICICHG
Minor Code       10 (0X000A)
Trace Groups     No groups assigned.
Trace Types       No types assigned.
Traced Parameters

%convtype = %d, out_file = %d, full_file = %d, HabPic = %d

PMPIC Major Code: 0X00C4 Minor Code: 1000 (0X03E8)
Description
PicPrint Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMPIC.PICPRINT

Minor Code
1000 (0X03E8)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
aram_str = %a, type = %d, filename = %a, hab = %d

PMGPI.DLL Trace Events

The tracepoints for the PMGPI.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:
Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMGPI Major Code: 0X00C5, sorted by minor code.
Trace events for PMGPI Major Code: 0X00C5, sorted by tracepoint.

Trace Events for PMGPI Major Code: 0X00C5, Sorted by Minor Code

00101 (0X0065) GpiCreatePS Pre-Invocation
00102 (0X0066) GpiDestroyPS Pre-Invocation
00103 (0X0067) GpiAssociate Pre-Invocation
00104 (0X0068) GpiRestorePS Pre-Invocation
00105 (0X0069) GpiSavePS Pre-Invocation
00106 (0X006A) GpiErase Pre-Invocation
00107 (0X006B) GpiQueryDevice Pre-Invocation
00108 (0X006C) GpiResetPS Pre-Invocation
00109 (0X006D) GpiSetPS Pre-Invocation
00110 (0X006E) GpiQueryPS Pre-Invocation
00111 (0X0070) GpiQueryTag Pre-Invocation
00112 (0X006F) GpiErrorSegmentData Pre-Invocation
00113 (0X0070) GpiQueryDrawControl Pre-Invocation
00114 (0X0071) GpiSetDrawControl Pre-Invocation
00115 (0X0072) GpiQueryDrawingMode Pre-Invocation
00116 (0X0073) GpiSetDrawingMode Pre-Invocation
00117 (0X0074) GpiQueryStopDraw Pre-Invocation
00118 (0X0075) GpiSetStopDraw Pre-Invocation
00201 (0X00C9) GpiCorrelateChain Pre-Invocation
00202 (0X00CA) GpiQueryTag Pre-Invocation
00203 (0X00CB) GpiSetTag Pre-Invocation
02316 (0X090C) MTPaintRegionMeta Pre-Invocation
02317 (0X090D) MTDevEscape Pre-Invocation
02318 (0X090E) MTRestorePS Pre-Invocation
02319 (0X090F) MTSavePS Pre-Invocation
02320 (0X0910) MTStartReadRequest Pre-Invocation
02321 (0X0911) MEndReadRequest Pre-Invocation
02322 (0X0912) MTStartWriteRequest Pre-Invocation
02323 (0X0913) MEndWriteRequest Pre-Invocation
02324 (0X0914) MTLongByteSwap Pre-Invocation

Trace Events for PMGPI Major Code: 0X00C5, Sorted by Tracepoint

GPIASSOCIATE 00103 (0X0067)
GPIBEGINAREA 01106 (0X0452)
GPIBEGINELEMENT 00500 (0X01F4)
GPIBEGINPATH 00800 (0X0320)
GPIBITBLT 01802 (0X070A)
GPIBOX 01212 (0X04BC)
GPIALLSEGMENTMATRIX 00605 (0X025D)
GPICHARSTRING 01416 (0X0588)
GPICHARSTRINGAT 01417 (0X0589)
GPICHARSTRINGPOS 01414 (0X0586)
GPICHARSTRINGPOSAT 01415 (0X0587)
GPICLOSEFIGURE 00802 (0X0322)
GPICLOSESEGMENT 00301 (0X012D)
GPICOMBINEFIGURE 00500 (0X01F4)
GPICOMMENT 01604 (0X0644)
GPICONVERT 00602 (0X025A)
GPICOPYMETAFILE 02104 (0X0838)
GPICORRELATECHAIN 00201 (0X00C9)
GPICORRELATEFROM 00210 (0X00D3)
GPICORRELATESEGMENT 00211 (0X00D3)
GPICREATEBITMAP 01804 (0X070C)
GPICREATELOGCOLORTABLE 00900 (0X0384)
GPICREATELOGFONT 01705 (0X06A9)
GPICREATEPS 00101 (0X0065)
GPICREATEFIGURE 01900 (0X076C)
GPIDELETEBITMAP 01800 (0X0708)
GPIDELETEELEMENT 00505 (0X01F9)
GPIDELETEELEMENTRANGE 00506 (0X01FA)
GPIDELETEELEMENTSBEETWEENLABELS 00507 (0X01FB)
GPIDELETEMETAFILE 02103 (0X0837)
GPIDELETESEGMENT 00302 (0X012E)
GPIDELETESEGMENTS 00509 (0X0135)
GPIDELETESETID 01700 (0X0644)
GPIDESTROYPSPS 00102 (0X0066)
GPIDESTROYPREGION 01902 (0X076E)
GPIDRAWCHAIN 00402 (0X0192)
GPIDRAWDYNAMICS 00405 (0X0195)
GPIDRAWFROM 00403 (0X0193)
GPIDRAWSEGMENT 00404 (0X0194)
GPIELEMENT 00503 (0X01F7)
GPIENDAREA 01107 (0X0453)
GPIENDELEMENT 00501 (0X0065)
GPIEQUALREGION 01904 (0X0770)
GPIEQUALPATH 00801 (0X01F4)
GPIEQUALREGION 01904 (0X01F5)
GPIEQUALPATH 00801 (0X0321)
GPIEREASE 01006 (0X006A)
GPIERRORSEGMENTDATA 00111 (0X006F)
GPIEXCLUDECLIPRECTANGLE 02004 (0X07D4)
GPIFILLPATH 00804 (0X0324)
GPIFULLARC 01303 (0X0517)
GPIGETDATA 00400 (0X0190)
GPIIMAGE 01600 (0X0640)
PMGPI Major Code: 0X00C5 Minor Code: 101 (0X0065)

**Description**
GpiCreatePS Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICREATEPS

**Minor Code**
101 (0X0065)

**Trace Groups**
No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%Options = %d, WidthHeight = %a, HDC = %d, Hab = %d
WidthHeight->cx = %d, WidthHeight->cy = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 102 (0X0066)

Description
GpiDestroyPS Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDESTROYPS

Minor Code
102 (0X0066)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 103 (0X0067)

Description
GpiAssociate Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIASSOCIATE

Minor Code
103 (0X0067)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%DcH = %d, GpiH = %d

--------------------------------------------
PMGPI Major Code: 0X00C5 Minor Code: 104 (0X0068)

Description: GpiRestorePS Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIRESTOREPS

Minor Code: 104 (0X0068)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Psid = %d, GpiH = %d

---------------------

PMGPI Major Code: 0X00C5 Minor Code: 105 (0X0069)

Description: GpiSavePS Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISAVEPS

Minor Code: 105 (0X0069)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%GpiH = %d

---------------------

PMGPI Major Code: 0X00C5 Minor Code: 106 (0X006A)

Description: GpiErase Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIERASE

Minor Code: 106 (0X006A)
PMGPI Major Code: 0X00C5 Minor Code: 107 (0X006B)

Description
GpiQueryDevice Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEVICE

Minor Code
107 (0X006B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 108 (0X006C)

Description
GpiResetPS Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIRESETPS

Minor Code
108 (0X006C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Psid = %d, GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 109 (0X006D)

Description: GpiSetPS Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETPS

Minor Code: 109 (0X006D)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Options = %d, PresentationSize = %a, GpiH = %d
PresentationSize->cx = %d, PresentationSize->cy = %d

PMGPI Major Code: 0X00C5 Minor Code: 110 (0X006E)

Description: GpiQueryPS Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPS

Minor Code: 110 (0X006E)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 111 (0X006F)

Description: GpiErrorSegmentData Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIERRORSEGMENTDATA
PMGPI Major Code: 0X00C5 Minor Code: 111 (0X006F)

Minor Code
111 (0X006F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 112 (0X0070)

Description
GpiQueryDrawControl Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDRAWCONTROL

Minor Code
112 (0X0070)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Control = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 113 (0X0071)

Description
GpiSetDrawControl Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETDRAWCONTROL

Minor Code
113 (0X0071)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Value = %d, Control = %d, GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 114 (0X0072)

Description GpiQueryDrawingMode Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDRAWINGMODE
Minor Code 114 (0X0072)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 115 (0X0073)

Description GpiSetDrawingMode Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPISETDRAWINGMODE
Minor Code 115 (0X0073)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

%Mode = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 116 (0X0074)

Description GpiQueryStopDraw Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSTOPDRAW

Minor Code
116 (0X0074)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 117 (0X0075)

Description
GpiSetStopDraw Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETSTOPDRAW

Minor Code
117 (0X0075)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Value = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 201 (0X00C9)

Description
GpiCorrelateChain Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATECHAIN

Minor Code
201 (0X00C9)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PMGPI Major Code: 0X00C5 Minor Code: 202 (0X00CA)

**Description**  
GpiQueryTag Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYTAG

**Minor Code**  
202 (0X00CA)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 203 (0X00CB)

**Description**  
GpiSetTag Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPISETTAG

**Minor Code**  
203 (0X00CB)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
%V = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 204 (0X00CC)
Description  GpiQueryPickApertureSize Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPICKAPERTURESIZE
Minor Code  204 (0X00CC)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 205 (0X00CD)

Description  GpiSetPickApertureSize Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPISETPICKAPERTURESIZE
Minor Code  205 (0X00CD)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%ApertureSize = %a, Options = %d, GpiH = %d
ApertureSize.cx = %d, ApertureSize.cy = %d

PMGPI Major Code: 0X00C5 Minor Code: 206 (0X00CE)

Description  GpiQueryPickAperturePosition Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPICKAPERTUREPOSITION
Minor Code  206 (0X00CE)
Trace Groups  No groups assigned.
Trace Types
No types assigned.

Traced Parameters

\%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 207 (0X00CF)

Description
GpiSetPickAperturePosition Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETPICKAPERTUREPOSITION

Minor Code
207 (0X00CF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%AperturePosition = %a, GpiH = %d
AperturePosition->x = %d, AperturePosition->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 208 (0X00D0)

Description
GpiQueryBoundaryData Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBOUNDARYDATA

Minor Code
208 (0X00D0)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 209 (0X00D1)
**Description**
GpiResetBoundaryData Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIRESETBOUNDARYDATA

**Minor Code**
209 (0X00D1)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 210 (0X00D2)

**Description**
GpiCorrelateFrom Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATEFROM

**Minor Code**
210 (0X00D2)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%MaxDepth = %d, MaxHits = %d, PickPosition = %a
Type = %d, LastSegment = %d, FirstSegment = %d, GpiH = %d
Pick->x = %d, Pick->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 211 (0X00D3)

**Description**
GpiCorrelateSegment Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATESEGMENT

**Minor Code**
Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%MaxDepth = %d, MaxHits = %d, PickPosition = %a
Type = %d, Segment = %d, GpiH = %d
Pick->x = %d, Pick->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 300 (0X012C)

Description
GpiOpenSegment Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIOPENSEGMENT

Minor Code
300 (0X012C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 301 (0X012D)

Description
GpiCloseSegment Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPICLOSESEGMENT

Minor Code
301 (0X012D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
PMGPI Major Code: 0X00C5 Minor Code: 302 (0X012E)

Description
GpiDeleteSegment Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDELETESEGMENT

Minor Code
302 (0X012E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

   %SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 303 (0X012F)

Description
GpiQueryInitialSegmentAttrs Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYINITIALSEGMENTATTRS

Minor Code
303 (0X012F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

   %Attribute = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 304 (0X0130)

Description
GpiSetInitialSegmentAttrs Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETINITIALSEGMENTATTRS

Minor Code: 304 (0X0130)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Value = %d, Attribute = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 305 (0X0131)

Description: GpiQuerySegmentAttrs Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTATTRS

Minor Code: 305 (0X0131)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Attribute = %d, SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 306 (0X0132)

Description: GpiSetSegmentAttrs Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTATTRS

Minor Code: 306 (0X0132)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:
PMGPI Major Code: 0X00C5 Minor Code: 307 (0X0133)

**Description**
GpiQuerySegmentPriority Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTPRIORITY

**Minor Code**
307 (0X0133)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Order = %d, ReferenceSegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 308 (0X0134)

**Description**
GpiSetSegmentPriority Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTPRIORITY

**Minor Code**
308 (0X0134)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Order = %d, ReferenceSegmentID = %d, SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 309 (0X0135)
Description: GpiDeleteSegments Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIDELETESEGMENTS
Minor Code: 309 (0X0135)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%LastSegment = %d, FirstSegment = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 310 (0X0136)

Description: GpiQuerySegmentNames Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTNAMES
Minor Code: 310 (0X0136)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%MaxNumber = %d, LastSegment = %d,
FirstSegment = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 400 (0X0190)

Description: GpiGetData Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIGETDATA
Minor Code: 400 (0X0190)
Trace Groups: No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%ReadLen = %d, Control = %d, Offset = %a,
SegmentName = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 401 (0X0191)

Description
GpiPutData Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPUTDATA
Minor Code
401 (0X0191)
Trace Groups
No groups assigned.
Trace Types
No types assigned.
Traced Parameters

%DPtr = %a, LenPtr = %d, Control = %d, GpiH = %d
%DPtr = %b%b %b%b %b%b %b%b %b%b

PMGPI Major Code: 0X00C5 Minor Code: 402 (0X0192)

Description
GpiDrawChain Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWCHAIN
Minor Code
402 (0X0192)
Trace Groups
No groups assigned.
Trace Types
No types assigned.
Traced Parameters

%GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 403 (0X0193)

Description  GpiDrawFrom Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWFROM
Minor Code  403 (0X0193)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%LastSegment = %d, FirstDegment = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 404 (0X0194)

Description  GpiDrawSegment Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWSEGMENT
Minor Code  404 (0X0194)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 405 (0X0195)

Description  GpiDrawDynamics Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWDYNAMICS
Minor Code
PMGPI Major Code: 0X00C5 Minor Code: 406 (0X0196)

Description  GpiRemoveDynamics Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIREMOVEDYNAMICS
Minor Code  406 (0X0196)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%LastSegmentID = %d, FirstSegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 500 (0X01F4)

Description  GpiBeginElement Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINELEMENT
Minor Code  500 (0X01F4)
Trace Groups  No groups assigned.
Trace Types  No types assigned.
Traced Parameters

%desc = %a, type = %d, GpiH = %d
desc-> = %s
PMGPI Major Code: 0X00C5 Minor Code: 501 (0X01F5)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiEndElement Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIENDELEMENT</td>
</tr>
<tr>
<td>Minor Code</td>
<td>501 (0X01F5)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 502 (0X01F6)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiLabel Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPILABEL</td>
</tr>
<tr>
<td>Minor Code</td>
<td>502 (0X01F6)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Label = %d, GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 503 (0X01F7)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiElement Pre-Invocation</th>
</tr>
</thead>
</table>
Public symbol defined dynamic tracepoint: PMGPI.GPIELEMENT

**Minor Code**
503 (0X01F7)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Data = %a, Length = %d, Desc = %a, Type = %d, GpiH = %d
Data = %s
Desc = %s

PMGPI Major Code: 0X00C5 Minor Code: 504 (0X01F8)

**Description**
GpiQueryElement Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENT

**Minor Code**
504 (0X01F8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%MaxLength = %d, Offset = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 505 (0X01F9)

**Description**
GpiDeleteElement Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENT

**Minor Code**
505 (0X01F9)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.
Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 506 (0X01FA)

Description
GpiDeleteElementRange Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENTRANGE

Minor Code
506 (0X01FA)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%LastElement = %d, FirstElement = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 507 (0X01FB)

Description
GpiDeleteElementsBetweenLabels Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENTSBETWEENLABELS

Minor Code
507 (0X01FB)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%LastLabel = %d, FirstLabel = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 508 (0X01FC)
GpiQueryEditMode Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYEDITMODE

Minor Code
508 (0X01FC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 509 (0X01FD)

GpiSetEditMode Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPISETEDITMODE

Minor Code
509 (0X01FD)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%EditMode = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 510 (0X01FE)

GpiQueryElementPointer Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENTPOINTER

Minor Code
510 (0X01FE)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

\%GpiH = \%d

PMGPI Major Code: 0X00C5 Minor Code: 511 (0X01FF)

Description
GpiSetElementPointer Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETELEMENTPOINTER

Minor Code
511 (0X01FF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%ElementNumber = \%d, GpiH = \%d

PMGPI Major Code: 0X00C5 Minor Code: 512 (0X0200)

Description
GpiOffsetElementPointer Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETELEMENTPOINTER

Minor Code
512 (0X0200)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

\%Offset = \%d, GpiH = \%d

PMGPI Major Code: 0X00C5 Minor Code: 513 (0X0201)
GpiQueryElementType Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENTTYPE

Minor Code 513 (0X0201)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%Data Length = %d
GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 514 (0X0202)

GpiSetElementPointerAtLabel Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPISETELEMENTPOINTERATLABEL

Minor Code 514 (0X0202)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%Label = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 600 (0X0258)

GpiQuerySegmentTransformMatrix Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTTRANSFORMMATRIX

Minor Code 600 (0X0258)

Trace Groups No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%Count = %d, SegmentID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 601 (0X0259)

Description
GpiSetSegmentTransformMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTTRANSFORMMATRIX

Minor Code
601 (0X0259)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Options = %d, Matrix = %a, Count = %d,
SegmentID = %d, GpiH = %d
Matrix->fxM11 = %d, Matrix->fxM12 = %d
Matrix->lM13 = %d, Matrix->fxM21 = %d
Matrix->fxM22 = %d, Matrix->lM23 = %d
Matrix->lM31 = %d, Matrix->lM32 = %d
Matrix->lM33 = %d

PMGPI Major Code: 0X00C5 Minor Code: 602 (0X025A)

Description
GpiConvert Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPICONVERT

Minor Code
602 (0X025A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

%d Count, %d TargetCoordSpace, %d SrcCoordSpace, %d GpiH

PMGPI Major Code: 0X00C5 Minor Code: 603 (0X025B)

Description
GpiQueryModelTransformMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMODELTRANSFORMMATRIX

Minor Code
603 (0X025B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%d Count, %d GpiH

PMGPI Major Code: 0X00C5 Minor Code: 604 (0X025C)

Description
GpiSetModelTransformMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETMODELTRANSFORMMATRIX

Minor Code
604 (0X025C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%d Options, %a Matrix, %d Count, %d GpiH
Matrix->fxM11 = %d, Matrix->fxM12 = %d
Matrix->lM13 = %d, Matrix->fxM21 = %d
Matrix->fxM22 = %d, Matrix->lM23 = %d
Matrix->lM31 = %d, Matrix->lM32 = %d
Matrix->lM33 = %d

---------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 605 (0X025D)

Description
GpiCallSegmentMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPICALLSEGMENTMATRIX

Minor Code
605 (0X025D)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Options = %d, Matrix = %a, Count = %d,
SegmentID = %d, GpiH = %d
Matrix->fxM11 = %d, Matrix->fxM12 = %d
Matrix->fxM13 = %d, Matrix->fxM21 = %d
Matrix->fxM22 = %d, Matrix->fxM23 = %d
Matrix->fxM31 = %d, Matrix->fxM32 = %d
Matrix->fxM33 = %d

---------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 606 (0X025E)

Description
GpiQueryDefaultViewMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEFAULTVIEWMATRIX

Minor Code
606 (0X025E)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Options = %d, Matrix = %a, Count = %d,
PMGPI Major Code: 0X00C5 Minor Code: 607 (0X025F)

Description
GpiSetDefaultViewMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETDEFAULTVIEWMATRIX

Minor Code
607 (0X025F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Options = %d, Matrix = %a, Count = %d, GpiH = %d
Matrix->fxM11 = %d, Matrix->fxM12 = %d
Matrix->fxM13 = %d, Matrix->fxM21 = %d
Matrix->fxM22 = %d, Matrix->fxM23 = %d
Matrix->fxM31 = %d, Matrix->fxM32 = %d
Matrix->fxM33 = %d

PMGPI Major Code: 0X00C5 Minor Code: 608 (0X0260)

Description
GpiQueryPageViewport Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPAGEVIEWPORT

Minor Code
608 (0X0260)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 609 (0X0261)

Description
GpiSetPageViewport Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETPAGEVIEWPORT

Minor Code
609 (0X0261)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Viewport = %a, GpiH = %d
Viewport->xLeft = %d, Viewport->yBottom = %d
Viewport->xRight = %d, Viewport->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 610 (0X0262)

Description
GpiQueryViewingTransformMatrix Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYVIEWINGTRANSFORMMATRIX

Minor Code
610 (0X0262)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Count = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 611 (0X0263)

Description
GpiSetViewingTransformMatrix Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETVIEWINGTRANSFORMMATRIX

Minor Code
611 (0X0263)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Options = %d, Matrix = %a, Count = %d, GpiH = %d
Matrix->fxM11 = %d, Matrix->fxM12 = %d
Matrix->lM13 = %d, Matrix->fxM21 = %d
Matrix->fxM22 = %d, Matrix->IM23 = %d
Matrix->IM31 = %d, Matrix->IM32 = %d
Matrix->IM33 = %d

PMGPI Major Code: 0X00C5 Minor Code: 700 (0X02BC)

Description
GpiSetGraphicsField Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETGRAPHICSFIELD

Minor Code
700 (0X02BC)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Field = %a, GpiH = %d
Field->xLeft = %d, Field->yBottom = %d
Field->xRight = %d, Field->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 701 (0X02BD)

Description
GpiQueryGraphicsField Pre-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYGRAPHICSFIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>701 (0X02BD)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 702 (0X02BE)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiSetViewingLimits Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPISETVIEWINGLIMITS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>702 (0X02BE)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Limits = %d, GpiH = %d</td>
</tr>
<tr>
<td></td>
<td>Limits-&gt;xLeft = %d, Limits-&gt;yBottom = %d</td>
</tr>
<tr>
<td></td>
<td>Limits-&gt;xRight = %d, Limits-&gt;yTop = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 703 (0X02BF)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiQueryViewingLimits Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYVIEWINGLIMITS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>703 (0X02BF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
</tbody>
</table>
Traced Parameters

%GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 800 (0X0320)

Description
GpiBeginPath Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINPATH

Minor Code
800 (0X0320)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%PathID = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 801 (0X0321)

Description
GpiEndPath Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIENDPATH

Minor Code
801 (0X0321)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 802 (0X0322)
**GpiCloseFigure Pre-Invocation**

Public symbol defined dynamic tracepoint: PMGPI.GPICLOSEFIGURE

**Minor Code**

802 (0X0322)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 803 (0X0323)

**GpiModifyPath Pre-Invocation**

Public symbol defined dynamic tracepoint: PMGPI.GPIMODIFYPATH

**Minor Code**

803 (0X0323)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Mode = %d, PathID = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 804 (0X0324)

**GpiFillPath Pre-Invocation**

Public symbol defined dynamic tracepoint: PMGPI.GPIFILLPATH

**Minor Code**

804 (0X0324)

**Trace Groups**

No groups assigned.
PMGPI Major Code: 0X00C5 Minor Code: 805 (0X0325)

Description
GpiSetClipPath Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETCLIPPATH

Minor Code
805 (0X0325)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Options = %d, PathID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 806 (0X0326)

Description
GpiStrokePath Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISTROKEPATH

Minor Code
806 (0X0326)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Options = %d, PathID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 900 (0X0384)
**Description**
GpiCreateLogColorTable Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICREATELOGCOLORTABLE

**Minor Code**
900 (0X0384)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Table=%a, Count=%d, Start=%d, Format=%d,
Options=%d, GpiH = %d
Table->field1 = %d, Table->field2 = %d
Table->field3 = %d, Table->field4 = %d

PMGPI Major Code: 0X00C5 Minor Code: 901 (0X0385)

**Description**
GpiRealizeColorTable Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIREALIZECOLORTABLE

**Minor Code**
901 (0X0385)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 902 (0X0386)

**Description**
GpiUnrealizeColorTable Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIUNREALIZECOLORTABLE
Minor Code: 902 (0X0386)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 903 (0X0387)

Description: GpiQueryColorData Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCOLORDATA

Minor Code: 903 (0X0387)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Count = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 904 (0X0388)

Description: GpiQueryLogColorTable Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLOGCOLORTABLE

Minor Code: 904 (0X0388)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Count = %d, Start = %d, Options = %d, GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 905 (0X0389)

Description  
GpiQueryRealColors Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREALCOLORS

Minor Code  
905 (0X0389)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%Count = %d, Start = %d, Options = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 906 (0X038A)

Description  
GpiQueryNearestColor Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYNEARESTCOLOR

Minor Code  
906 (0X038A)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%RequiredColor = %d, Options = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 907 (0X038B)

Description  
GpiQueryColorIndex Pre-Invocation
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCOLORINDEX

**Minor Code**
907 (0X038B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

```
%RGBColor = %d, Options = %d, GpiH = %d
```

PMGPI Major Code: 0X00C5 Minor Code: 908 (0X038C)

**Description**
GpiQueryRGBColor Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYRGBCOLOR

**Minor Code**
908 (0X038C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

```
%ColorIndex = %d, Options = %d, GpiH = %d
```

PMGPI Major Code: 0X00C5 Minor Code: 1000 (0X03E8)

**Description**
GpiSetColor Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETCOLOR

**Minor Code**
1000 (0X03E8)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

```
%ColorIndex = %d, Options = %d, GpiH = %d
```
<table>
<thead>
<tr>
<th>Description</th>
<th>PMGPI Major Code: 0X00C5 Minor Code: 1001 (0X3E9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>GpiQueryColor Pre-Invocation</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1001 (0X3E9)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>PMGPI Major Code: 0X00C5 Minor Code: 1002 (0X3EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>GpiSetBackColor Pre-Invocation</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1002 (0X3EA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Color = %d, GpiH = %d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>PMGPI Major Code: 0X00C5 Minor Code: 1003 (0X3EB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>GpiQueryBackColor Pre-Invocation</td>
</tr>
</tbody>
</table>

Tracepoint: PMGPI.GPIQUERYBACKCOLOR
Minor Code: 1003 (0X03EB)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1004 (0X03EC)

Description: GpiSetMix Pre-Invocation
Tracepoint: PMGPI.GPISETMIX
Minor Code: 1004 (0X03EC)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
%MixMode = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1005 (0X03ED)

Description: GpiQueryMix Pre-Invocation
Tracepoint: PMGPI.GPIQUERYMIX
Minor Code: 1005 (0X03ED)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:
PMGPI Major Code: 0X00C5 Minor Code: 1006 (0X03EE)

Description: GpiSetBackMix Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETBACKMIX
Minor Code: 1006 (0X03EE)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

PMGPI Major Code: 0X00C5 Minor Code: 1007 (0X03EF)

Description: GpiQueryBackMix Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBACKMIX
Minor Code: 1007 (0X03EF)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

PMGPI Major Code: 0X00C5 Minor Code: 1008 (0X03F0)

Description: GpiSetAttrs Pre-Invocation
<table>
<thead>
<tr>
<th>Tracepoint</th>
<th>Public symbol defined dynamic tracepoint: PMGPI.GPISETATTRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor Code</td>
<td>1008 (0x03F0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%DefMask = %d, AttrMask = %d, PrimType = %d, GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0x00C5 Minor Code: 1009 (0x03F1)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiQueryAttrs Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYATTRS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1009 (0x03F1)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%AttrMask = %d, PrimType = %d, GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0x00C5 Minor Code: 1010 (0x03F2)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiSetAttrMode Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPISETATTRMODE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1010 (0x03F2)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td></td>
</tr>
</tbody>
</table>
PMGPI Major Code: 0X00C5 Minor Code: 1011 (0X03F3)

Description
GpiQueryAttrMode Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYATTRMODE

Minor Code
1011 (0X03F3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1100 (0X044C)

Description
GpiSetPatternSet Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERNSET

Minor Code
1100 (0X044C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%PatternSetID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1101 (0X044D)

Description
GpiQueryPatternSet Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERNSET

**Minor Code**
1101 (0X044D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%GpiH = %d

------------------------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1102 (0X044E)

**Description**
GpiSetPatternRefPoint Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERNREFPOINT

**Minor Code**
1102 (0X044E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%PatternRefPt = %a, GpiH = %d
PatternRefPt->x = %d, PatternRefPt->y = %d

------------------------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1103 (0X044F)

**Description**
GpiQueryPatternRefPoint Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERNREFPOINT

**Minor Code**
1103 (0X044F)

**Trace Groups**
No groups assigned.

**Trace Types**
PMGPI Major Code: 0X00C5 Minor Code: 1104 (0X0450)

Description
GpiSetPattern Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERN

Minor Code
1104 (0X0450)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%PatternSymbol = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1105 (0X0451)

Description
GpiQueryPattern Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERN

Minor Code
1105 (0X0451)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1106 (0X0452)
GpiBeginArea Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINAREA

Minor Code 1106 (0X0452)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

%AreaOption = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1107 (0X0453)

GpiEndArea Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIENDAREA

Minor Code 1107 (0X0453)
Trace Groups No groups assigned.
Trace Types No types assigned.
Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1200 (0X04B0)

GpiSetLineType Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPISETLINETYPE

Minor Code 1200 (0X04B0)
Trace Groups No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%LineType = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1201 (0X04B1)

Description
GpiQueryLineType Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINETYPE

Minor Code
1201 (0X04B1)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1202 (0X04B2)

Description
GpiSetLineWidth Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEWIDTH

Minor Code
1202 (0X04B2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%LineWidth = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1203 (0X04B3)
Description GpiQueryLineWidth Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEWIDTH

Minor Code 1203 (0X04B3)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1204 (0X04B4)

Description GpiSetLineWidthGeom Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIServiceLINEWIDTHGEOM

Minor Code 1204 (0X04B4)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%LineWidthGeom = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1205 (0X04B5)

Description GpiQueryLineWidthGeom Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEWIDTHGEOM

Minor Code 1205 (0X04B5)

Trace Groups No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1206 (0X04B6)

Description
GpiSetLineEnd Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEEND

Minor Code
1206 (0X04B6)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%LineEnd = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1207 (0X04B7)

Description
GpiQueryLineEnd Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEEND

Minor Code
1207 (0X04B7)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1208 (0X04B8)
<table>
<thead>
<tr>
<th>Description</th>
<th>GpiSetLineJoin Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEJOIN</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1208 (0X04B8)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%LineJoin = %d, GpiH = %d</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>PMGPI Major Code: 0X00C5 Minor Code: 1209 (0X04B9)</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>GpiQueryLineJoin Pre-Invocation</td>
</tr>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEJOIN</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1209 (0X04B9)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>PMGPI Major Code: 0X00C5 Minor Code: 1210 (0X04BA)</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>GpiSetCurrentPosition Pre-Invocation</td>
</tr>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPISETCURRENTPOSITION</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1210 (0X04BA)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMGPI Major Code: 0X00C5 Minor Code: 1211 (0X04BB)

**Description**
GpiQueryCurrentPosition Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCURRENTPOSITION

**Minor Code**
1211 (0X04BB)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
\%
\%

PMGPI Major Code: 0X00C5 Minor Code: 1212 (0X04BC)

**Description**
GpiBox Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIBOX

**Minor Code**
1212 (0X04BC)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

\%
\%
\%
\%
\%
\%
\%
PMGPI Major Code: 0X00C5 Minor Code: 1213 (0X04BD)

**Description**
GpiMove Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIMOVE

**Minor Code**
1213 (0X04BD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %Point = %a, GpiH = %d
- Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1214 (0X04BE)

**Description**
GpiLine Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPILINE

**Minor Code**
1214 (0X04BE)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %Point = %a, GpiH = %d
- Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1215 (0X04BF)
Description: GpiPolyLine Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYLINE

Minor Code: 1215 (0X04BF)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\%
Point = %a, Count = %d, GpiH = %d

Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1300 (0X0514)

Description: GpiSetArcParams Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETARCPARAMS

Minor Code: 1300 (0X0514)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

\%
ArcParams = %a, GpiH = %d

ArcParams->IP = %d, ArcParams->IQ = %d

ArcParams->IR = %d, ArcParams->IS = %d

PMGPI Major Code: 0X00C5 Minor Code: 1301 (0X0515)

Description: GpiQueryArcParams Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYARCPARAMS

Minor Code: 1301 (0X0515)
**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1302 (0X0516)

**Description**
GpiPointArc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIPOINTARC

**Minor Code**
1302 (0X0516)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Point = %a, GpiH = %d
Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1303 (0X0517)

**Description**
GpiFullArc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIFULLARC

**Minor Code**
1303 (0X0517)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Multiplier = %d, Control = %d, GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 1304 (0X0518)

**Description**
GpiPartialArc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIPARTIALARC

**Minor Code**
1304 (0X0518)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %SweepAngle=%d, StartAngle=%d, Multiplier=%d,
- Centre=%a, GpiH = %d
- Centre->x = %d, Centre->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1305 (0X0519)

**Description**
GpiPolyFilletSharp Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYFILLETSHARP

**Minor Code**
1305 (0X0519)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %Sharpness = %a, Point = %a, Count = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1306 (0X051A)

**Description**
GpiPolySpline Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYSPLINE

Minor Code
1306 (0X051A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Point = %a, Count = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1307 (0X051B)

Description
GpiPolyFillet Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYFILLET

Minor Code
1307 (0X051B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Point = %a, Count = %d, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1400 (0X0578)

Description
GpiQueryTextBox Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYTEXTBOX

Minor Code
1400 (0X0578)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Point = %a, Count = %d, GpiH = %d

--------------------------------------------
PMGPI Major Code: 0X00C5 Minor Code: 1401 (0X0579)

**Description**
GpiQueryDefCharBox Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEFCHARBOX

**Minor Code**
1401 (0X0579)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1402 (0X057A)

**Description**
GpiSetCharSet Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARSET

**Minor Code**
1402 (0X057A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Lcid = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1403 (0X057B)
GpiQueryCharSet Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSET

Minor Code
1403 (0X057B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1404 (0X057C)

GpiSetCharBox Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARBOX

Minor Code
1404 (0X057C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Size = %a, GpiH = %d
Size->cx = %d, Size->cy

PMGPI Major Code: 0X00C5 Minor Code: 1405 (0X057D)

GpiQueryCharBox Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARBOX

Minor Code
1405 (0X057D)

Trace Groups
No groups assigned.
PMGPI Major Code: 0X00C5 Minor Code: 1406 (0X057E)

**Description**
GpiSetCharAngle Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARANGLE

**Minor Code**
1406 (0X057E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

```
%Gradient = %a, GpiH = %d
Gradient->x = %d, Gradient->y = %d
```

PMGPI Major Code: 0X00C5 Minor Code: 1407 (0X057F)

**Description**
GpiQueryCharAngle Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARANGLE

**Minor Code**
1407 (0X057F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

```
%GpiH = %d
```

PMGPI Major Code: 0X00C5 Minor Code: 1408 (0X0580)
PMGPI Major Code: 0X00C5 Minor Code: 1410 (0X0582)
Description: GpiSetCharDirection Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARDIRECTION
Minor Code: 1410 (0X0582)
PMGPI Major Code: 0X00C5 Minor Code: 1411 (0X0583)

Description
GpiQueryCharDirection Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARDIRECTION

Minor Code
1411 (0X0583)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Direction = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1412 (0X0584)

Description
GpiSetCharMode Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARMODE

Minor Code
1412 (0X0584)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Mode = %d, GpiH = %d
### PMGPI Major Code: 0X00C5 Minor Code: 1413 (0X0585)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiQueryCharMode Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARMODE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1413 (0X0585)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
</tbody>
</table>

### PMGPI Major Code: 0X00C5 Minor Code: 1414 (0X0586)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiCharStringPos Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGPOS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1414 (0X0586)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Vector=%a, String=%a, Count=%d, Options=%d, Rect=%a, GpiH = %d, String = %s, Rect-&gt;xLeft = %d, Rect-&gt;yBottom = %d, Rect-&gt;xRight = %d, Rect-&gt;yTop = %d</td>
</tr>
</tbody>
</table>

### PMGPI Major Code: 0X00C5 Minor Code: 1415 (0X0587)
**Description**  
GpiCharStringPosAt Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGPOSAT

**Minor Code**  
1415 (0X0587)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%Vector=%a, String=%a, Count=%d, Options=%d,  
Rect=%a, Point=%a, GpiH = %d  
String = %s  
Rect->xLeft = %d, Rect->yBottom = %d  
Rect->xRight = %d, Rect->yTop = %d  
Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1416 (0X0588)

**Description**  
GpiCharString Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRING

**Minor Code**  
1416 (0X0588)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%String = %a, Count = %d, GpiH = %d  
String = %s

PMGPI Major Code: 0X00C5 Minor Code: 1417 (0X0589)

**Description**  
GpiCharStringAt Pre-Invocation
Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGAT

Minor Code  
1417 (0X0589)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%String = %a, Count = %d, Point = %a, GpiH = %d
String = %s
Point->x = %d, Point->y = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1418 (0X058A)

Description  
GpiQueryCharStringPos Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSTRINGPOS

Minor Code  
1418 (0X058A)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%String = %a, Count = %d, Options = %d, GpiH = %d
String = %s

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1419 (0X058B)

Description  
GpiQueryCharStringPosAt Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSTRINGPOSAT

Minor Code  
1419 (0X058B)

Trace Groups
PMGPI Major Code: 0X00C5 Minor Code: 1500 (0X05DC)

**Description**
GpiSetMarkerSet Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKERSET

**Minor Code**
1500 (0X05DC)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%MarkerSetID = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1501 (0X05DD)

**Description**
GpiQueryMarkerSet Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKERSET

**Minor Code**
1501 (0X05DD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%MarkerSetID = %d, GpiH = %d
PMGPI Major Code: 0X00C5 Minor Code: 1502 (0X05DE)

Description  
GpiSetMarker Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKER

Minor Code  
1502 (0X05DE)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%MarkerSymbol = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1503 (0X05DF)

Description  
GpiQueryMarker Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKER

Minor Code  
1503 (0X05DF)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1504 (0X05E0)

Description  
GpiSetMarkerBox Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKERBOX

Minor Code 1504 (0X05E0)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%MarkerBoxSize = %a, GpiH = %d
MarkerBoxSize->cx = %d, MarkerBoxSize->cy = %d

PMGPI Major Code: 0X00C5 Minor Code: 1505 (0X05E1)

Description GpiQueryMarkerBox Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKERBOX

Minor Code 1505 (0X05E1)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters

%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1506 (0X05E2)

Description GpiMarker Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMGPI.GPIMARKER

Minor Code 1506 (0X05E2)

Trace Groups No groups assigned.

Trace Types No types assigned.
Traced Parameters

%Point = %a, GpiH = %d
Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1507 (0X05E3)

Description
GpiPolyMarker Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYMARKER

Minor Code
1507 (0X05E3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Point = %a, Count = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1600 (0X0640)

Description
GpiImage Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIIMAGE

Minor Code
1600 (0X0640)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Data = %a, Length = %d, ImageSize = %a,
Format = %d, GpiH = %d
%Data = %b%b %b%b %b%b %b%b %b%b%b%b
ImageSize->cx = %d, ImageSize->cy = %d
PMGPI Major Code: 0X00C5 Minor Code: 1601 (0X0641)

Description
GpiPop Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPOP

Minor Code
1601 (0X0641)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Count = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1602 (0X0642)

Description
GpiPtVisible Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIPTVISIBLE

Minor Code
1602 (0X0642)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Point = %a, GpiH = %d
Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1603 (0X0643)

Description
GpiRectVisible Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIRECTVISIBLE

Minor Code
1603 (0X0643)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Rect = %a, GpiH = %d
Rect->xLeft = %d, Rect->yBottom = %d
Rect->xRight = %d, Rect->yTop = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1604 (0X0644)

Description
GpiComment Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPICOMMENT

Minor Code
1604 (0X0644)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Comment = %a, Count = %d, GpiH = %d
Comment = %s

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1700 (0X06A4)

Description
GpiDeleteSetId Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPDELETESETID

Minor Code
1700 (0X06A4)

Trace Groups
No groups assigned.
PMGPI Major Code: 0X00C5 Minor Code: 1701 (0X06A5)

Description       GpiQueryNumberSetIds Pre-Invocation
Tracepoint        Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYNUMBERSETIDS
Minor Code        1701 (0X06A5)
Trace Groups      No groups assigned.
Trace Types       No types assigned.
Traced Parameters %GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1702 (0X06A6)

Description       GpiQuerySetIds Pre-Invocation
Tracepoint        Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSETIDS
Minor Code        1702 (0X06A6)
Trace Groups      No groups assigned.
Trace Types       No types assigned.
Traced Parameters %GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1703 (0X06A7)
Description: GpiLoadFonts Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPILOADFONTS

Minor Code: 1703 (0X06A7)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

```
%Filename = %a, GpiH = %d
Filename = %s
```

PMGPI Major Code: 0X00C5 Minor Code: 1704 (0X06A8)

Description: GpiUnloadFonts Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIUNLOADFONTS

Minor Code: 1704 (0X06A8)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

```
%Filename = %a, GpiH = %d
Filename = %s
```

PMGPI Major Code: 0X00C5 Minor Code: 1705 (0X06A9)

Description: GpiCreateLogFont Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPICREATELOGFONT

Minor Code:
PMGPI Major Code: 0X00C5 Minor Code: 1706 (0X06AA)

Description: GpiQueryFonts Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTS
Minor Code: 1706 (0X06AA)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
Traced Parameters:

%Lcid = %d, FontName = %a, GpiH = %d
FontName = %s

PMGPI Major Code: 0X00C5 Minor Code: 1707 (0X06AB)

Description: GpiQueryFontMetrics Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTMETRICS
Minor Code: 1707 (0X06AB)
Trace Groups: No groups assigned.
Trace Types: No types assigned.
**Traced Parameters**

%Length = %d, GpiH = %d

--------------------------------------------

**PMGPI Major Code: 0X00C5 Minor Code: 1708 (0X06AC)**

**Description**
GpiQueryKerningPairs Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYKERNINGPAIRS

**Minor Code**
1708 (0X06AC)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%KerningPairs = %d, GpiH = %d

--------------------------------------------

**PMGPI Major Code: 0X00C5 Minor Code: 1709 (0X06AD)**

**Description**
GpiQueryWidthTable Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYWIDTHTABLE

**Minor Code**
1709 (0X06AD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Count = %d, FirstChar = %d, GpiH = %d

--------------------------------------------

**PMGPI Major Code: 0X00C5 Minor Code: 1710 (0X06AE)**
<table>
<thead>
<tr>
<th>Description</th>
<th>GpiSetCp Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPISETCP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1710 (0X06AE)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%CodePage = %w, GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 1711 (0X06AF)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiQueryCp Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1711 (0X06AF)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%GpiH = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 1712 (0X06B0)

<table>
<thead>
<tr>
<th>Description</th>
<th>GpiQueryFontFileDescriptions Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTFILEDESCRIPTIONS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>1712 (0X06B0)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMGPI Major Code: 0X00C5 Minor Code: 1800 (0X0708)

Description
GpiDeleteBitmap Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEBITMAP

Minor Code
1800 (0X0708)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HBitmap = %a, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1801 (0X0709)

Description
GpiSetBitmap Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAP

Minor Code
1801 (0X0709)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HBitmap = %a, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1802 (0X070A)
**GpiBitBlt Pre-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIBITBLT

**Minor Code**
1802 (0X070A)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
\%
Options = %d, Rop = %d, Pointl = %a,
Count = %d, GpiH = %d

--------------------------------------------

**PMGPI Major Code: 0X00C5 Minor Code: 1803 (0X070B)**

**GpiWCBitBlt Pre-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIWCBITBLT

**Minor Code**
1803 (0X070B)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
\%
Options = %d, Rop = %d, Pointl = %a,
Count = %d, GpiH = %d

--------------------------------------------

**PMGPI Major Code: 0X00C5 Minor Code: 1804 (0X070C)**

**GpiCreateBitmap Pre-Invocation**

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICREATEBITMAP

**Minor Code**
1804 (0X070C)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Data = %a, Options = %d, GpiH = %d
Data = %s

PMGPI Major Code: 0X00C5 Minor Code: 1805 (0X070D)

**Description**
GpiSetBitmapDimension Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPDIMENSION

**Minor Code**
1805 (0X070D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Hbitmap = %d, SizeL = %a
SizeL->cx = %d, SizeL->cy = %d

PMGPI Major Code: 0X00C5 Minor Code: 1806 (0X070E)

**Description**
GpiQueryBitmapDimension Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPDIMENSION

**Minor Code**
1806 (0X070E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
PMGPI Major Code: 0X00C5 Minor Code: 1807 (0X070F)

Description
GpiQueryDeviceBitmapFormats Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEVICEBITMAPFORMATS

Minor Code
1807 (0X070F)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Count = %a, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1808 (0X0710)

Description
GpiQueryBitmapParameters Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPPARAMETERS

Minor Code
1808 (0X0710)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HBitmap = %a

PMGPI Major Code: 0X00C5 Minor Code: 1809 (0X0711)

Description
GpiQueryBitmapBits Pre-Invocation
Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPBITS

Minor Code
1809 (0X0711)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Long = %d, Long = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1810 (0X0712)

Description
GpiSetBitmapBits Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPBITS

Minor Code
1810 (0X0712)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Data = %a, Scans = %d, ScanStart = %d, GpiH = %d
Data = %s

PMGPI Major Code: 0X00C5 Minor Code: 1811 (0X0713)

Description
GpiSetPel Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETPEL

Minor Code
1811 (0X0713)

Trace Groups
No groups assigned.

Trace Types
No types assigned.
Traced Parameters

%Point = %a, GpiH = %d
%Point->x = %d, Point->y

PMGPI Major Code: 0X00C5 Minor Code: 1812 (0X0714)

Description
GpiQueryPel Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPEL

Minor Code
1812 (0X0714)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%HPS = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1813 (0X0715)

Description
GpiSetBitmapId Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPID

Minor Code
1813 (0X0715)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Lcid = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1814 (0X0716)
Description: GpiQueryBitmapHandle Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPHANDLE

Minor Code: 1814 (0X0716)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Lcid = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1900 (0X076C)

Description: GpiCreateRegion Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPICREATEREGION

Minor Code: 1900 (0X076C)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Rct = %a, Count = %d, GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 1901 (0X076D)

Description: GpiSetRegion Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPISETREGION

Minor Code:
1901 (0X076D)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Rct = %a, Count = %d, Hrgn = %a, GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1902 (0X076E)

**Description**
GpiDestroyRegion Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIDESTROYREGION

**Minor Code**
1902 (0X076E)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%Hrgn = %a, GpiH = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1903 (0X076F)

**Description**
GpiCombineRegion Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPICOMBINEREGION

**Minor Code**
1903 (0X076F)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
PMGPI Major Code: 0X00C5 Minor Code: 1904 (0X0770)

Description
GpiEqualRegion Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIEQUALREGION

Minor Code
1904 (0X0770)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Src2 = %a, Src1 = %a, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 1905 (0X0771)

Description
GpiOffsetRegion Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETREGION

Minor Code
1905 (0X0771)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Point = %a, Hrgn = %a, GpiH = %d
%Point->x = %d, Pointl->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1906 (0X0772)
Description: GpiPtInRegion Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIPTINREGION

Minor Code: 1906 (0X0772)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Pointl = %a, Hrgn = %a, GpiH = %d
Pointl->x = %d, Pointl->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 1907 (0X0773)

Description: GpiRectInRegion Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIRECTINREGION

Minor Code: 1907 (0X0773)

Trace Groups: No groups assigned.

Trace Types: No types assigned.

Traced Parameters:

%Rct = %a, Hrgn = %a, GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop   = %d

PMGPI Major Code: 0X00C5 Minor Code: 1908 (0X0774)

Description: GpiQueryRegionBox Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREGIONBOX

Minor Code: 1908 (0X0774)
Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters

%Hrgn = %a, GpiH = %d

---------------------------------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1909 (0X0775)

Description  GpiQueryRegionRects Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREGIONRECTS

Minor Code  1909 (0X0775)

Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters

%Rct = %a, Hrgn = %a, GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop = %d

---------------------------------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 1910 (0X0776)

Description  GpiPaintRegion Pre-Invocation

Tracepoint  Public symbol defined dynamic tracepoint: PMGPI.GPIPAINTREGION

Minor Code  1910 (0X0776)

Trace Groups  No groups assigned.

Trace Types  No types assigned.

Traced Parameters
%HRgn = %a, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2000 (0X07D0)

**Description**
GpiSetClipRegion  Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISETCLIPREGION

**Minor Code**
2000 (0X07D0)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%HRgn = %a, GpiH = %d

---

PMGPI Major Code: 0X00C5 Minor Code: 2001 (0X07D1)

**Description**
GpiQueryClipRegion  Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCLIPREGION

**Minor Code**
2001 (0X07D1)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%Hps = %d, GpiH = %d

---

PMGPI Major Code: 0X00C5 Minor Code: 2002 (0X07D2)

**Description**
GpiQueryClipBox  Pre-Invocation
PMGPI Major Code: 0X00C5 Minor Code: 2002 (0X07D2)

Description
GpiQueryClipBox Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPQUERYCLIPBOX

Minor Code
2002 (0X07D2)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2003 (0X07D3)

Description
GpiIntersectClipRectangle Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIINTERSECTCLIPRECTANGLE

Minor Code
2003 (0X07D3)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 2004 (0X07D4)

Description
GpiExcludeClipRectangle Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIEXCLUDELIPRECTANGLE

Minor Code
2004 (0X07D4)

Trace Groups
No groups assigned.

Trace Types
Traced Parameters

%Rct = %a, GpiH = %d
Rct->xLeft = %d, Rct->yBottom = %d
Rct->xRight = %d, Rct->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 2005 (0X07D5)

Description
GpiOffsetClipRegion  Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETCLIPREGION

Minor Code
2005 (0X07D5)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Point = %a, GpiH = %d
Point->x = %d, Point->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 2100 (0X0834)

Description
GpiLoadMetaFile  Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPILOADMETAFILE

Minor Code
2100 (0X0834)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%Filename = %a, GpiH = %d
Filename = %s
PMGPI Major Code: 0X00C5 Minor Code: 2101 (0X0835)

**Description**
GpiPlayMetaFile Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPIPLAYMETAFILE

**Minor Code**
2101 (0X0835)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %Count2 = %d, GpiH = %d
- OptArray = %a, Count1 = %d
- OptArray = %s

PMGPI Major Code: 0X00C5 Minor Code: 2102 (0X0836)

**Description**
GpiSaveMetaFile Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.GPISAVEMETAFILE

**Minor Code**
2102 (0X0836)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

- %Filename = %a, GpiH = %d
- Filename = %s

PMGPI Major Code: 0X00C5 Minor Code: 2103 (0X0837)
**GpiDeleteMetaFile** Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEMETAFILE

**Minor Code**

2103 (0X0837)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMF = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2104 (0X0838)

**GpiCopyMetaFile** Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPICOPYMETAFILE

**Minor Code**

2104 (0X0838)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMF = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2105 (0X0839)

**GpiQueryMetaFileLength** Pre-Invocation

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMETAFILELENGTH

**Minor Code**

2105 (0X0839)

**Trace Groups**

No groups assigned.
Trace Types
No types assigned.

Traced Parameters
%HMF = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2106 (0X083A)

Description
GpiQueryMetaFileBits  Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMETAFILEBITS

Minor Code
2106 (0X083A)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Length = %d, Offset = %d, GpiH = %d

PMGPI Major Code: 0X00C5 Minor Code: 2107 (0X083B)

Description
GpiSetMetaFileBits  Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.GPISETMETAFILEBITS

Minor Code
2107 (0X083B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Buffer = %a, Length = %d, Offset = %d, GpiH = %d

Buffer = %s

PMGPI Major Code: 0X00C5 Minor Code: 2300 (0X08FC)
**PMGPI Major Code: 0X00C5 Minor Code: 2300 (0X08FC)**

**Description**
MTIDPStoreMeta Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTIDPSTOREMETA

**Minor Code**
2300 (0X08FC)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%Long = %d, Word = %w, Pbyte = %a, HDC = %d

**PMGPI Major Code: 0X00C5 Minor Code: 2301 (0X08FD)**

**Description**
MTEnableKerningMeta Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTENABLEKERNINGMETA

**Minor Code**
2301 (0X08FD)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**
%Long = %d, HDC = %d

**PMGPI Major Code: 0X00C5 Minor Code: 2302 (0X08FE)**

**Description**
MTDisplayFlagMeta Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTDISPLAYFLAGMETA

**Minor Code**
2302 (0X08FE)

**Trace Groups**
No groups assigned.
Trace Types
No types assigned.

Traced Parameters

%Bool = %d, HDC = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 2303 (0X08FF)

Description
MTCreateLogColorTableMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTCREATELOGCOLORTABLEMETA

Minor Code
2303 (0X08FF)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%ULong = %d, ULong = %d, ULong = %d, ULong = %d, HDC = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 2304 (0X0900)

Description
MTSetCodePageMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTSETCODEPAGEMETA

Minor Code
2304 (0X0900)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters

%ULong = %d, HDC = %d

--------------------------------------------

PMGPI Major Code: 0X00C5 Minor Code: 2305 (0X0901)
<table>
<thead>
<tr>
<th>Description</th>
<th>MTDeleteSetIDMeta Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTDELETESETIDMETA</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>2305 (0x0901)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>%ULong = %d, HDC = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 2306 (0X0902)

<table>
<thead>
<tr>
<th>Description</th>
<th>MTSetGraphicsFieldMeta Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTSETGRAPHICSFIELDMETA</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>2306 (0X0902)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>No groups assigned.</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>No types assigned.</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>%Rect = %a, HDC = %d</td>
</tr>
<tr>
<td></td>
<td>Rect-&gt;xLeft = %d, Rect-&gt;yBottom = %d</td>
</tr>
<tr>
<td></td>
<td>Rect-&gt;xRight = %d, Rect-&gt;yTop = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 2307 (0X0903)

<table>
<thead>
<tr>
<th>Description</th>
<th>MTRestMeta Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTRESETMETA</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td></td>
</tr>
</tbody>
</table>
PMGPI Major Code: 0X00C5 Minor Code: 2307 (0X0903)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%ULong = %d, HDC = %d

PMGPI Major Code: 0X00C5 Minor Code: 2308 (0X0904)

Description
MTEraseMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTERASEMETA

Minor Code
2308 (0X0904)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HDC = %d

PMGPI Major Code: 0X00C5 Minor Code: 2309 (0X0905)

Description
MTAssociateMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTASSOCIATEMETA

Minor Code
2309 (0X0905)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HDC = %d
PMGPI Major Code: 0X00C5 Minor Code: 2310 (0X0906)

Description  
MTVerifyPageUnits Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.MTVERIFYPAGEUNITS

Minor Code  
2310 (0X0906)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters

%ULong = %d, HDC = %d

PMGPI Major Code: 0X00C5 Minor Code: 2311 (0X0907)

Description  
MTBitBltMeta Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMGPI.MTBITBLTMETA

Minor Code  
2311 (0X0907)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters

%Long = %d, Long = %d, Pointl = %a, HDC1= %d, HDC2= %d
Pointl->x = %d, Pointl->y = %d

PMGPI Major Code: 0X00C5 Minor Code: 2312 (0X0908)

Description  
MTWCBitBltMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTWCBITBLTMETA

**Minor Code:** 2312 (0X0908)

**Trace Groups:** No groups assigned.

**Trace Types:** No types assigned.

**Traced Parameters:**

```
%HBITMAP = %d, HDC = %d
```

--------------------------------------------

**PMGPI Major Code:** 0X00C5 **Minor Code:** 2313 (0X0909)

**Description:** MTSetPelMeta Pre-Invocation

**Tracepoint:** Public symbol defined dynamic tracepoint: PMGPI.MTSETPELMETA

**Minor Code:** 2313 (0X0909)

**Trace Groups:** No groups assigned.

**Trace Types:** No types assigned.

**Traced Parameters:**

```
%Pointl = %a, HDC = %d
Pointl->x = %d, Pointl->y = %d
```

--------------------------------------------

**PMGPI Major Code:** 0X00C5 **Minor Code:** 2314 (0X090A)

**Description:** MTSelectClipMeta Pre-Invocation

**Tracepoint:** Public symbol defined dynamic tracepoint: PMGPI.MTSELECTCLIPMETA

**Minor Code:** 2314 (0X090A)

**Trace Groups:** No groups assigned.

**Trace Types:** No types assigned.

**Traced Parameters:**
PMGPI Major Code: 0X00C5 Minor Code: 2315 (0X090B)

Description
MTClipRegionMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.HTCLIPREGIONMETA

Minor Code
2315 (0X090B)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Hrgn = %d, HDC = %d

Rect->xLeft = %d, Rect->yBottom = %d
Rect->xRight = %d, Rect->yTop = %d

PMGPI Major Code: 0X00C5 Minor Code: 2316 (0X090C)

Description
MTPaintRegionMeta Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTPAINTREGIONMETA

Minor Code
2316 (0X090C)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%Hrgn = %d, HDC = %d

PMGPI Major Code: 0X00C5 Minor Code: 2317 (0X090D)
<table>
<thead>
<tr>
<th>Description</th>
<th>MTDDevEscape Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTDEVESCAPE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2317 (0X090D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Long = %d, Long = %d, HDC = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 2318 (0X090E)

<table>
<thead>
<tr>
<th>Description</th>
<th>MTRestorePS Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTRESTOREPS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2318 (0X090E)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
<tr>
<td>Trace Types</td>
<td>No types assigned.</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>%Long = %d, HDC = %d</td>
</tr>
</tbody>
</table>

PMGPI Major Code: 0X00C5 Minor Code: 2319 (0X090F)

<table>
<thead>
<tr>
<th>Description</th>
<th>MTSavePS Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMGPI.MTSAVEPS</td>
</tr>
<tr>
<td>Minor Code</td>
<td>2319 (0X090F)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>No groups assigned.</td>
</tr>
</tbody>
</table>
PMGPI Major Code: 0X00C5 Minor Code: 2320 (0X0910)

Description
MTStartReadRequest Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTSTARTREADREQUEST

Minor Code
2320 (0X0910)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HDC = %d

PMGPI Major Code: 0X00C5 Minor Code: 2321 (0X0911)

Description
MTEndReadRequest Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMGPI.MTENDREADREQUEST

Minor Code
2321 (0X0911)

Trace Groups
No groups assigned.

Trace Types
No types assigned.

Traced Parameters
%HMF = %d

PMGPI Major Code: 0X00C5 Minor Code: 2322 (0X0912)
**Description**
MTStartWriteRequest Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTSTARTWRITEREQUEST

**Minor Code**
2322 (0X0912)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HMF = %d

PMGPI Major Code: 0X00C5 Minor Code: 2323 (0X0913)

**Description**
MTEndWriteRequest Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTENDWRITEREQUEST

**Minor Code**
2323 (0X0913)

**Trace Groups**
No groups assigned.

**Trace Types**
No types assigned.

**Traced Parameters**

%HMF = %d

PMGPI Major Code: 0X00C5 Minor Code: 2324 (0X0914)

**Description**
MTLongByteSwap Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMGPI.MTLONGBYTESWAP

**Minor Code**
2324 (0X0914)

**Trace Groups**
No groups assigned.
PMSPL.DLL Trace Events

The trace points for the PMSPL.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

<table>
<thead>
<tr>
<th>Trace events for PMSPL Major Code: 0X00C6, sorted by minor code.</th>
<th>Trace events for PMSPL Major Code: 0X00C6, sorted by tracepoint.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Trace Events for PMSPL Major Code: 0X00C6, Sorted by Minor Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00002 (0X0002) SpiControlDevice Pre-Invocation</td>
</tr>
<tr>
<td>00003 (0X0003) SpiQueryDevice Pre-Invocation</td>
</tr>
<tr>
<td>00004 (0X0004) SpiEnumDevice Pre-Invocation</td>
</tr>
<tr>
<td>00005 (0X0005) SpiCreateDevice Pre-Invocation</td>
</tr>
<tr>
<td>00006 (0X0006) SpiSetDevice Pre-Invocation</td>
</tr>
<tr>
<td>00007 (0X0007) SpiDeleteDevice Pre-Invocation</td>
</tr>
<tr>
<td>00008 (0X0008) SpiReleaseJob Pre-Invocation</td>
</tr>
<tr>
<td>00009 (0X0009) SpiHoldJob Pre-Invocation</td>
</tr>
<tr>
<td>00010 (0X000A) SpiDeleteJob Pre-Invocation</td>
</tr>
<tr>
<td>00011 (0X000B) DosPrintJobSchedule Pre-Invocation</td>
</tr>
<tr>
<td>00012 (0X000C) DosPrintJobAdd Pre-Invocation</td>
</tr>
<tr>
<td>00013 (0X000D) DosPrintJobAdd2 Pre-Invocation</td>
</tr>
<tr>
<td>00014 (0X000E) SpiQueryJob Pre-Invocation</td>
</tr>
<tr>
<td>00015 (0X000F) SpiSetJob Pre-Invocation</td>
</tr>
<tr>
<td>00016 (0X0010) SpiEnumJob Pre-Invocation</td>
</tr>
<tr>
<td>00017 (0X0011) SpiCreateQueue Pre-Invocation</td>
</tr>
<tr>
<td>00018 (0X0012) SpiHoldQueue Pre-Invocation</td>
</tr>
<tr>
<td>00019 (0X0013) SpiReleaseQueue Pre-Invocation</td>
</tr>
<tr>
<td>00020 (0X0014) SpiDeleteQueue Pre-Invocation</td>
</tr>
<tr>
<td>00021 (0X0015) SpiPurgeQueue Pre-Invocation</td>
</tr>
<tr>
<td>00022 (0X0016) SpiQueryQueue Pre-Invocation</td>
</tr>
<tr>
<td>00023 (0X0017) SpiSetQueue Pre-Invocation</td>
</tr>
<tr>
<td>00024 (0X0018) SpiEnumQueue Pre-Invocation</td>
</tr>
<tr>
<td>00025 (0X0019) SpiEnumDriver Pre-Invocation</td>
</tr>
<tr>
<td>00026 (0X001A) SpiEnumQueueProcessor Pre-Invocation</td>
</tr>
<tr>
<td>00027 (0X001B) SpiEnumPort Pre-Invocation</td>
</tr>
<tr>
<td>00028 (0X001C) DosPrintJobGetId Pre-Invocation</td>
</tr>
<tr>
<td>00029 (0X001D) Spi32PrmSpool Pre-Invocation</td>
</tr>
<tr>
<td>00030 (0X001E) SpiQueryDriver Pre-Invocation</td>
</tr>
<tr>
<td>00031 (0X001F) SpiSetDriver Pre-Invocation</td>
</tr>
<tr>
<td>00032 (0X0020) SpiCopyJob Pre-Invocation</td>
</tr>
<tr>
<td>00033 (0X0021) SpiQueryJobFile Pre-Invocation</td>
</tr>
<tr>
<td>00034 (0X0022) SpiEnumPrinterSpiEnumQueue Pre-Invocation</td>
</tr>
<tr>
<td>00049 (0X0031) Spi32QmOpen Pre-Invocation</td>
</tr>
</tbody>
</table>
00050 (0X0032) Spl32QmStartDoc Pre-Invocation
00051 (0X0033) Spl32QmWrite Pre-Invocation
00052 (0X0034) Spl32QmWriteFile Pre-Invocation
00053 (0X0035) Spl32QmEndDoc Pre-Invocation
00054 (0X0036) Spl32QmAbortDoc Pre-Invocation
00055 (0X0037) Spl32QmClose Pre-Invocation
00056 (0X0038) Spl32QmAbort Pre-Invocation
00057 (0X0039) Spl32QmQueryPinfo Pre-Invocation
00058 (0X003A) Spl32QmSetStatus Pre-Invocation
00059 (0X003B) Spl32QmSetup Pre-Invocation
00081 (0X0051) Spl32MessageBox Pre-Invocation
00082 (0X0052) Prt32Open Pre-Invocation
00083 (0X0053) Prt32Write Pre-Invocation
00084 (0X0054) Prt32DevIOCtl Pre-Invocation
00085 (0X0055) Prt32Close Pre-Invocation
00086 (0X0056) Prt32Abort Pre-Invocation
00087 (0X0057) PrtNewPage Pre-Invocation
00088 (0X0058) PrtResetAbort Pre-Invocation
00089 (0X0059) PrtAbortDoc Pre-Invocation
01113 (0X0071) Spl32StdOpen Pre-Invocation
01113 (0X0071) Spl32StdOpen Pre-Invocation
01114 (0X0072) Spl32StdClose Pre-Invocation
01114 (0X0072) Spl32StdClose Pre-Invocation
01115 (0X0073) Spl32StdStart Pre-Invocation
01116 (0X0074) Spl32StdStop Pre-Invocation
01117 (0X0075) Spl32StdQueryLength Pre-Invocation
01118 (0X0076) Spl32StdGetInfo Pre-Invocation
01119 (0X0077) Spl32StdDelete Pre-Invocation
02256 (0X0100) SplFSOpen Pre-Invocation
02257 (0X0101) SplFSFirstWrite Pre-Invocation
02258 (0X0102) SplFSWriteFail Pre-Invocation
02259 (0X0103) SplFSClose Pre-Invocation
02260 (0X0104) SplFSSetTitle Pre-Invocation
02261 (0X0105) SplFSActCP Pre-Invocation
02262 (0X0106) SplFSVerifyCP Pre-Invocation
02263 (0X0107) SplFSReturnCPAct Pre-Invocation
02264 (0X0108) AttachPort Pre-Invocation
02265 (0X0109) DetachPort Pre-Invocation
03004 (0X0130) PrintDestControl Pre-Invocation
03005 (0X0131) PrintDestGetInfo Pre-Invocation
03006 (0X0132) PrintDestEnum Pre-Invocation
03007 (0X0133) PrintDestAdd Pre-Invocation
03008 (0X0134) PrintDestSetInfo Pre-Invocation
03009 (0X0135) PrintDestDel Pre-Invocation
03010 (0X0136) PrintJobContinue Pre-Invocation
03011 (0X0141) PrintJobPause Pre-Invocation
03012 (0X0142) PrintJobDel Pre-Invocation
03013 (0X0143) PrintJobSchedule Pre-Invocation
03024 (0X0144) PrintJobAdd Pre-Invocation
03025 (0X0145) PrintJobGetInfo Pre-Invocation
03026 (0X0146) PrintJobSetInfo Pre-Invocation
03027 (0X0147) PrintJobEnum Pre-Invocation
03030 (0X0160) PrintQPurge Pre-Invocation
03037 (0X0151) PrintQPurge Pre-Invocation
03038 (0X0152) PrintQContinue Pre-Invocation
03039 (0X0153) PrintQAdd Pre-Invocation
03040 (0X0154) PrintQDel Pre-Invocation
03041 (0X0155) PrintQGetInfo Pre-Invocation
03042 (0X0156) PrintQSetInfo Pre-Invocation
03043 (0X0157) PrintQEnum Pre-Invocation
03044 (0X0158) PrintDriverEnum Pre-Invocation
03045 (0X0159) PrintQProcessorEnum Pre-Invocation
03046 (0X015A) PrintPortEnum Pre-Invocation
03068 (0X0170) SpiWarning Pre-Invocation
03068 (0X0170) SpiRWarning Pre-Invocation
03069 (0X0171) SpiError Pre-Invocation
03069 (0X0171) SpiRError Pre-Invocation
03070 (0X0172) SpiPanic Pre-Invocation
03070 (0X0172) SpiRPanic Pre-Invocation
03071 (0X0173) SpiEnNotRunning Pre-Invocation
03072 (0X0174) SpiEnNoMemory Pre-Invocation
03073 (0X0175) SpiLogWarning Pre-Invocation
03074 (0X0176) SpiLogError Pre-Invocation
Trace Events for PMSPL Major Code: 0X00C6, Sorted by Tracepoint

- AttachPort 00264 (0X0108)
- AttachPort 33032 (0X8108)
- DetachPort 00265 (0X0109)
- DetachPort 33033 (0X8109)
- DosPrintJobAdd 00012 (0X000C)
- DosPrintJobAdd 32780 (0X800C)
- DosPrintJobAdd2 00013 (0X000D)
- DosPrintJobAdd2 32781 (0X800D)
- DosPrintJobGetId 00028 (0X001C)
- DosPrintJobGetId 32796 (0X801C)
- DosPrintJobSchedule 00011 (0X000B)
- DosPrintJobSchedule 32779 (0X800B)
- GetNextId 33034 (0X810A)
- PRT32OPEN 00082 (0X0052)
- PRT32WRITE 00083 (0X0053)
- PostRefreshMsg 00384 (0X0180)
- PrintDestAdd 00307 (0X8133)
- PrintDestAdd 33075 (0X8133)
- PrintDestAdd2 00013 (0X000D)
- PrintDestAdd2 32781 (0X800D)
- PrintDestGetId 00304 (0X0130)
- PrintDestGetId 33072 (0X8130)
- PrintDestControl 00304 (0X0130)
- PrintDestControl 33072 (0X8130)
- PrintDestDel 000309 (0X0135)
- PrintDestDel 33077 (0X8135)
- PrintDestEnum 00306 (0X0132)
- PrintDestEnum 33074 (0X8132)
- PrintDestGetInfo 00305 (0X0131)
- PrintDestGetInfo 33073 (0X8131)
- PrintDestGetInfo 00305 (0X0131)
- PrintDestGetInfo 33073 (0X8131)
- PrintDestSetInfo 00308 (0X0134)
- PrintDestSetInfo 33076 (0X8134)
- PrintDriverEnum 00344 (0X0158)
- PrintDriverEnum 33112 (0X8158)
- PrintJobAdd 000324 (0X0144)
- PrintJobAdd 33092 (0X8144)
- PrintJobContinue 00320 (0X0140)
- PrintJobContinue 33088 (0X8140)
- PrintJobDel 00322 (0X0142)
- PrintJobDel 33090 (0X8142)
PrintJobEnum 00327 (0X0147)
PrintJobEnum 33095 (0X8147)
PrintJobGetInfo 00325 (0X0145)
PrintJobGetInfo 33093 (0X8145)
PrintJobPause 00321 (0X0141)
PrintJobPause 33089 (0X8141)
PrintJobSchedule 00323 (0X0143)
PrintJobSchedule 33091 (0X8143)
PrintJobSetInfo 00326 (0X0146)
PrintJobSetInfo 33094 (0X8146)
PrintPortEnum 00346 (0X015A)
PrintPortEnum 33114 (0X815A)
PrintQAdd 00339 (0X0153)
PrintQAdd 33107 (0X8153)
PrintQContinue 00338 (0X0152)
PrintQContinue 33106 (0X8152)
PrintQDel 00340 (0X0154)
PrintQDel 33108 (0X8154)
PrintQEnum 00343 (0X0157)
PrintQEnum 33111 (0X8157)
PrintQGetInfo 00341 (0X0155)
PrintQGetInfo 33109 (0X8155)
PrintQPause 00336 (0X0150)
PrintQPause 33104 (0X8150)
PrintQProcessorEnum 00345 (0X0159)
PrintQProcessorEnum 33113 (0X8159)
PrintQPurge 00337 (0X0151)
PrintQPurge 33105 (0X8151)
PrintQSetInfo 00342 (0X0156)
PrintQSetInfo 33110 (0X8156)
Prt32Abort 00086 (0X0056)
Prt32Close 00085 (0X0055)
Prt32Close 32853 (0X8055)
Prt32DevIOCtl 00084 (0X0054)
Prt32DevIOCtl 32852 (0X8054)
Prt32Open 32850 (0X8052)
Prt32Write 32851 (0X8053)
PrtAbortDoc 00089 (0X0059)
PrtAbortDoc 32857 (0X8059)
PrtNewPage 00087 (0X0057)
PrtNewPage 32855 (0X8057)
PrtResetAbort 00088 (0X0058)
PrtResetAbort 32856 (0X8058)
SPL32STDCLOSE 00114 (0X0072)
SPL32STDELETE 00119 (0X0077)
SPL32STDOOPEN 00113 (0X0071)
SPL32STQUERYLENGTH 00117 (0X0075)
SPL32STDDOSTART 00115 (0X0073)
SPL32STDDOSTOP 00116 (0X0074)
Spl32MessageBox 00081 (0X0051)
Spl32MessageBox 32849 (0X8051)
Spl32PrmSpool 00029 (0X001D)
Spl32PrmSpool 32797 (0X801D)
Spl32QmAbort 00056 (0X0038)
Spl32QmAbort 32824 (0X8038)
Spl32QmAbortDoc 00054 (0X0036)
Spl32QmAbortDoc 32822 (0X8036)
Spl32QmClose 00055 (0X0037)
Spl32QmClose 32823 (0X8037)
Spl32QmEndDoc 00053 (0X0035)
Spl32QmEndDoc 32821 (0X8035)
Spl32QmOpen 00049 (0X0031)
Spl32QmOpen 32817 (0X8031)
Spl32QmQueryPinfo 00057 (0X0039)
Spl32QmQueryPinfo 32825 (0X8039)
Spl32QmSetStatus 00058 (0X003A)
Spl32QmSetStatus 32826 (0X803A)
Spl32QmSetup 00059 (0X003B)
Spl32QmSetup 32827 (0X803B)
Spl32QmStartDoc 00050 (0X0032)
Spl32QmStartDoc 32818 (0X8032)
Spl32QmWrite 00051 (0X0033)
PMSPL Major Code: 0X00C6 Minor Code: 2 (0X0002)

**Description**
SplControlDevice Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplControlDevice

**Minor Code**
2 (0X0002)

**Trace Groups**
DOS

**Trace Types**
PRE

**Traced Parameters**
ulControl=%F, PortName=%s, ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 3 (0X0003)

**Description**
SplQueryDevice Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplQueryDevice

**Minor Code**
3 (0X0003)

**Trace Groups**
DOS
Trace Types
PRE

Traced Parameters
ulLevel=%F, cbBuf=%F, pszPrintDeviceName=%s
ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 4 (0X0004)

Description
SplEnumDevice Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplEnumDevice

Minor Code
4 (0X0004)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 5 (0X0005)

Description
SplCreateDevice Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplCreateDevice

Minor Code
5 (0X0005)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------
PMSPL Major Code: 0X00C6 Minor Code: 6 (0X0006)

**Description**  
SplSetDevice Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplSetDevice

**Minor Code**  
6 (0X0006)

**Trace Groups**  
DOS

**Trace Types**  
PRE

**Traced Parameters**

- pszPrinter=%s, ulParmNum=%F, ulLevel=%F, cbBuf=%F
- ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 7 (0X0007)

**Description**  
SplDeleteDevice Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplDeleteDevice

**Minor Code**  
7 (0X0007)

**Trace Groups**  
DOS

**Trace Types**  
PRE

**Traced Parameters**

- pszPrinter=%s, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 8 (0X0008)

**Description**  
SplReleaseJob Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplReleaseJob
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>8 (0X0008)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>ulJob=%F, ComputerName=%s, QueueName=%s</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 9 (0X0009)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplHoldJob Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplHoldJob</td>
</tr>
<tr>
<td>Minor Code</td>
<td>9 (0X0009)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>ulJob=%F, ComputerName=%s, QueueName=%s</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 10 (0X000A)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplDeleteJob Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplDeleteJob</td>
</tr>
<tr>
<td>Minor Code</td>
<td>10 (0X000A)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>ulJob=%F, ComputerName=%s, QueueName=%s</td>
</tr>
</tbody>
</table>
PMSPL Major Code: 0X00C6 Minor Code: 11 (0X000B)

Description
DosPrintJobSchedule Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobSchedule

Minor Code
11 (0X000B)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
UlJob=%w, ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 12 (0X000C)

Description
DosPrintJobAdd Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd

Minor Code
12 (0X000C)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
QueueName=%s, cbBuf=%w, pBuf=%#f
ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 13 (0X000D)

Description
DosPrintJobAdd2 Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd2

| Minor Code | 13 (0x000D) |
| Trace Groups | DOS |
| Trace Types | PRE |
| Traced Parameters | QueueName=%s, ulLevel=%w, cbBuf=%w, pBuf=%r%F, ComputerName=%s |

--------------------------------------------

PMSPL Major Code: 0x00C6 Minor Code: 14 (0x000E)

| Description | SplQueryJob Pre-Invocation |
| Tracepoint | Public symbol defined dynamic tracepoint: PMSPL.SplQueryJob |
| Minor Code | 14 (0x000E) |
| Trace Groups | DOS |
| Trace Types | PRE |
| Traced Parameters | ulJob=%F, ulLevel=%F, cbBuf=%F, ComputerName=%s, QueueName=%s |

--------------------------------------------

PMSPL Major Code: 0x00C6 Minor Code: 15 (0x000F)

| Description | SplSetJob Pre-Invocation |
| Tracepoint | Public symbol defined dynamic tracepoint: PMSPL.SplSetJob |
| Minor Code | 15 (0x000F) |
| Trace Groups | DOS |
| Trace Types | PRE |
Traced Parameters

ulJob=%F, ulParmNum=%F, ulLevel=%F, cbBuf=%F, pBuf=%u
ComputerName=%s QueueName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 16 (0X0010)

Description
SplEnumJob Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplEnumJob

Minor Code
16 (0X0010)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters

QueueName=%s, ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 17 (0X0011)

Description
SplCreateQueue Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplCreateQueue

Minor Code
17 (0X0011)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters

ulLevel=%F, cbBuf=%F, pbBuf=%r%F
ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 18 (0X0012)
<table>
<thead>
<tr>
<th>Description</th>
<th>SplHoldQueue Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplHoldQueue</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>18 (0X0012)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>QueueName=%s, ComputerName=%s</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 19 (0X0013)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplReleaseQueue Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplReleaseQueue</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>19 (0X0013)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>QueueName=%s, ComputerName=%s</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 20 (0X0014)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplDeleteQueue Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplDeleteQueue</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>20 (0X0014)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
</tbody>
</table>
Traces Types
PRE

Traced Parameters
QueueName=%s, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 21 (0X0015)

Description
SplPurgeQueue Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplPurgeQueue

Minor Code
21 (0X0015)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
QueueName=%s, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 22 (0X0016)

Description
SplQueryQueue Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplQueryQueue

Minor Code
22 (0X0016)

Trace Groups
DOS

Trace Types
PRE

Traced Parameters
QueueName=%s, ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 23 (0X0017)
**Description**  
SplSetQueue Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplSetQueue

**Minor Code**  
23 (0X0017)

**Trace Groups**  
DOS

**Trace Types**  
PRE

**Traced Parameters**

QueueName=%s, ulParmNum=%F, ulLevel=%F, cbBuf=%F, pBuf=%r%F

ComputerName=%s

--------------------------------------------

**PMSPL Major Code: 0X00C6 Minor Code: 24 (0X0018)**

**Description**  
SplEnumQueue Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueue

**Minor Code**  
24 (0X0018)

**Trace Groups**  
DOS

**Trace Types**  
PRE

**Traced Parameters**

ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

**PMSPL Major Code: 0X00C6 Minor Code: 25 (0X0019)**

**Description**  
SplEnumDriver Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplEnumDriver

**Minor Code**  
25 (0X0019)
**Trace Groups**
DOS

**Trace Types**
PRE

**Traced Parameters**
ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

**PMSPL Major Code: 0X00C6 Minor Code: 26 (0X001A)**

**Description**
SplEnumQueueProcessor Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueueProcessor

**Minor Code**
26 (0X001A)

**Trace Groups**
DOS

**Trace Types**
PRE

**Traced Parameters**
ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------

**PMSPL Major Code: 0X00C6 Minor Code: 27 (0X001B)**

**Description**
SplEnumPort Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplEnumPort

**Minor Code**
27 (0X001B)

**Trace Groups**
DOS

**Trace Types**
PRE

**Traced Parameters**
ulLevel=%F, cbBuf=%F, ComputerName=%s

--------------------------------------------
### PMSPL Major Code: 0X00C6 Minor Code: 28 (0X001C)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>DosPrintJobGetId Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobGetId</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>28 (0X001C)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>hFile=%F, cbInfo=%F</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 29 (0X001D)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Spl32PrmSpool Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32PrmSpool</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>29 (0X001D)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>PRE</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>pszInLogAddr=%s, pszOutLogAddr=%s</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 30 (0X001E)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>SpiQueryDriver Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SpiQueryDriver</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>30 (0X001E)</td>
</tr>
</tbody>
</table>
Trace Groups: DOS
Trace Types: PRE
Traced Parameters:
ulLevel=%F, cbBuf=%F, DriverName=%s, PrinterName=%s, ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 31 (0X001F)
Description: SplSetDriver Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplSetDriver
Minor Code: 31 (0X001F)
Trace Groups: DOS
Trace Types: PRE
Traced Parameters:
ulLevel=%F, ulParmNum=%F, cbBuf=%F, DriverName=%s, PrinterName=%s, ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 32 (0X0020)
Description: SplCopyJob Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplCopyJob
Minor Code: 32 (0X0020)
Trace Groups: DOS
Trace Types: PRE
Traced Parameters:
SourceJobID=%F, srcQueue=%s, srcComputer=%s
TargetQueue=%s, TargetComputer=%s

PMSPL Major Code: 0X00C6 Minor Code: 33 (0X0021)

- **Description**: SplQueryJobFile Pre-Invocation
- **Tracepoint**: Public symbol defined dynamic tracepoint: PMSPL.SplQueryJobFile
- **Minor Code**: 33 (0X0021)
- **Trace Groups**: DOS
- **Trace Types**: PRE
- **Traced Parameters**:
  - JobID=%F, Queue=%s, Computer=%s

PMSPL Major Code: 0X00C6 Minor Code: 34 (0X0022)

- **Description**: SplEnumPrinter SplEnumQueue Pre-Invocation
- **Tracepoint**: Public symbol defined dynamic tracepoint: PMSPL.SplEnumPrinter
- **Minor Code**: 34 (0X0022)
- **Trace Groups**: DOS
- **Trace Types**: PRE
- **Traced Parameters**:
  - uLevel=%F, flType=%F, cbBuf=%F, ComputerName=%s

PMSPL Major Code: 0X00C6 Minor Code: 49 (0X0031)

- **Description**
Spl32QmOpen Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmOpen

**Minor Code**
49 (0X0031)

**Trace Groups**
SPLQM

**Trace Types**
PRE

**Traced Parameters**

cData =%F pQMOPENDATA=%F pszLogAddress=%F
pszDriverName=%F pdriv=%F pszDataType=%F
pszComment=%F pszQueueProcName=%F pszQProcParams=%F
pszSpoolerParams=%F pszNetworkParams=%F
LogAddr-%s DriverName-%s DataType-%s
pdriv->cb=%F pdriv->lVersion=%F pdriv->szDeviceName-%s
SpoolerParams-%s Comment-%s QProc-%s QProcParms-%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 50 (0X0032)

**Description**
Spl32QmStartDoc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmStartDoc

**Minor Code**
50 (0X0032)

**Trace Groups**
SPLQM

**Trace Types**
PRE

**Traced Parameters**

hSpl=%F, pszDocument=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 51 (0X0033)

**Description**
Spl32QmWrite Pre-Invocation
Tracepoint  
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWrite

Minor Code  
51 (0X0033)

Trace Groups  
SPLQM

Trace Types  
PRE

Traced Parameters  

hSpl=%F cbDataIn=%F pbData=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 52 (0X0034)

Description  
Spl32QmWriteFile Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWriteFile

Minor Code  
52 (0X0034)

Trace Groups  
SPLQM

Trace Types  
PRE

Traced Parameters  

hSpl=%F Filename to write=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 53 (0X0035)

Description  
Spl32QmEndDoc Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmEndDoc

Minor Code  
53 (0X0035)

Trace Groups  
SPLQM

Trace Types  
PRE

Traced Parameters  

PMSPL Major Code: 0X00C6 Minor Code: 54 (0X0036)

**Description**
Spl32QmAbortDoc Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbortDoc

**Minor Code**
54 (0X0036)

**Trace Groups**
SPLQM

**Trace Types**
PRE

**Traced Parameters**

hSpl =%F

PMSPL Major Code: 0X00C6 Minor Code: 55 (0X0037)

**Description**
Spl32QmClose Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmClose

**Minor Code**
55 (0X0037)

**Trace Groups**
SPLQM

**Trace Types**
PRE

**Traced Parameters**

hSpl =%F

PMSPL Major Code: 0X00C6 Minor Code: 56 (0X0038)

**Description**
Spl32QmAbort Pre-Invocation
Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbort

Minor Code

56 (0X0038)

Trace Groups

SPLQM

Trace Types

PRE

Traced Parameters

hSpl =%F

PMSPL Major Code: 0X00C6 Minor Code: 57 (0X0039)

Description

Spl32QmQueryPinfo Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmQueryPinfo

Minor Code

57 (0X0039)

Trace Groups

SPLQM

Trace Types

PRE

Traced Parameters

Logical address=%s

PMSPL Major Code: 0X00C6 Minor Code: 58 (0X003A)

Description

Spl32QmSetStatus Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetStatus

Minor Code

58 (0X003A)

Trace Groups

SPLQM

Trace Types

PRE

Traced Parameters
PMSPL Major Code: 0X00C6 Minor Code: 59 (0X003B)

Description
Spl32QmSetup Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetup

Minor Code
59 (0X003B)

Trace Groups
SPLQM

Trace Types
PRE

Traced Parameters
Type=%F DopData-%F %F %F %F
LogAddr-%s DriverName-%s DataType-%s
pdriv->cb=%F pdriv->lVersion=%F pdriv->szDeviceName-%s

PMSPL Major Code: 0X00C6 Minor Code: 81 (0X0051)

Description
Spl32MessageBox Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.Spl32MessageBox

Minor Code
81 (0X0051)

Trace Groups
PRT

Trace Types
PRE

Traced Parameters
fErrInfo=%F, fErrData=%F, fStyle=%F, LogAddr=%s
Caption=%s, Text=%s

PMSPL Major Code: 0X00C6 Minor Code: 82 (0X0052)
<table>
<thead>
<tr>
<th>Description</th>
<th>Prt32Open Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.PRT32OPEN</td>
</tr>
<tr>
<td>Minor Code</td>
<td>82 (0X0052)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRT</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>pszDeviceName=%s, openFlag=%F, openMode=%F, reserved=%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 83 (0X0053)

<table>
<thead>
<tr>
<th>Description</th>
<th>Prt32Write Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.PRT32WRITE</td>
</tr>
<tr>
<td>Minor Code</td>
<td>83 (0X0053)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRT</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>hFile=%F, pchData=%F, cbData=%F, *pchData=%r%b</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 84 (0X0054)

<table>
<thead>
<tr>
<th>Description</th>
<th>Prt32DevlOCtl Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Prt32DevlOCtl</td>
</tr>
<tr>
<td>Minor Code</td>
<td>84 (0X0054)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRT</td>
</tr>
</tbody>
</table>
PMSPL Major Code: 0X00C6 Minor Code: 85 (0X0055)

**Description**
Prt32Close Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Prt32Close

**Minor Code**
85 (0X0055)

**Trace Groups**
PRT

**Trace Types**
PRE

**Traced Parameters**
hFile=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 86 (0X0056)

**Description**
Prt32Abort Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.Prt32Abort

**Minor Code**
86 (0X0056)

**Trace Groups**
PRT

**Trace Types**
PRE

**Traced Parameters**
hFile=%F

--------------------------------------------
PMSPL Major Code: 0X00C6 Minor Code: 87 (0X0057)

Description: PrtNewPage Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrtNewPage
Minor Code: 87 (0X0057)
Trace Groups: PRT
Trace Types: PRE
Traced Parameters:

hFile=%F ulPageNumber=%F

--------------------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 88 (0X0058)

Description: PrtResetAbort Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrtResetAbort
Minor Code: 88 (0X0058)
Trace Groups: PRT
Trace Types: PRE
Traced Parameters:

hFile=%F

--------------------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 89 (0X0059)

Description: PrtAbortDoc Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrtAbortDoc
Minor Code: 89 (0X0059)
Trace Groups: PRT
Trace Types: PRE
Traced Parameters:

hFile=%F, pchData=%F, cbData=%F, ulFlags=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 113 (0X0071)

Description: Spl32StdOpen Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SPL32STDOPEN
Minor Code: 113 (0X0071)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 113 (0X0071)

Description: Spl32StdOpen Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdOpen
Minor Code: 113 (0X0071)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F
PMSPL Major Code: 0X00C6 Minor Code: 114 (0X0072)

Description: Spl32StdClose Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SPL32STDCLOSE
Minor Code: 114 (0X0072)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 114 (0X0072)

Description: Spl32StdClose Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdClose
Minor Code: 114 (0X0072)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 115 (0X0073)

Description: Spl32StdStart Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SPL32STDSTART
Minor Code:
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F

PMSPL Major Code: 0X00C6 Minor Code: 116 (0X0074)

Description: Spl32StdStop Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SPL32STDSTOP
Minor Code: 116 (0X0074)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hDC=%F

PMSPL Major Code: 0X00C6 Minor Code: 117 (0X0075)

Description: Spl32StdQueryLength Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SPL32STDQUERYLENGTH
Minor Code: 117 (0X0075)
Trace Groups: STD
Trace Types: PRE
Traced Parameters:

hMetaFile=%F
PMSPL Major Code: 0X00C6 Minor Code: 118 (0X0076)

**Description**  
Spl32StdGetBits Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.Spl32StdGetBits

**Minor Code**  
118 (0X0076)

**Trace Groups**  
STD

**Trace Types**  
PRE

**Traced Parameters**

hMetaFile=%F, offData=%F, cbData=%F, pchData=%F

PMSPL Major Code: 0X00C6 Minor Code: 119 (0X0077)

**Description**  
Spl32StdDelete Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SPL32STDDELETE

**Minor Code**  
119 (0X0077)

**Trace Groups**  
STD

**Trace Types**  
PRE

**Traced Parameters**

hMetaFile=%F

PMSPL Major Code: 0X00C6 Minor Code: 256 (0X0100)

**Description**  
SplFSOpen Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplFSOpen
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>256 (0x0100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>uHandle=%w Codepage=%w, Key=%F, Port=%s</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 257 (0X0101)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSFirstWrite Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSFirstWrite</td>
</tr>
<tr>
<td>Minor Code</td>
<td>257 (0x0101)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>uHandle=%w</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 258 (0X0102)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSWriteFail Pre-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSWriteFail</td>
</tr>
<tr>
<td>Minor Code</td>
<td>258 (0x0102)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>PRE</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>uHandle=%w, ErrorCode=%w</td>
</tr>
</tbody>
</table>
PMSPL Major Code: 0X00C6 Minor Code: 259 (0X0103)

Description
SplFSClose Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplFSClose

Minor Code
259 (0X0103)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
uHandle=%w

PMSPL Major Code: 0X00C6 Minor Code: 260 (0X0104)

Description
SplFSSetTitle Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplFSSetTitle

Minor Code
260 (0X0104)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
uHandle=%w, Title=%s

PMSPL Major Code: 0X00C6 Minor Code: 261 (0X0105)

Description
SplFSActCP Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplFSActCP
Minor Code: 261 (0x0105)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: uHandle=%w, CodePage=%w

--------------------------------------------

PMSPL Major Code: 0x00C6 Minor Code: 262 (0x0106)

Description: SplFSVerifyCP Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplFSVerifyCP
Minor Code: 262 (0x0106)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: uHandle=%w, CodePage=%w

--------------------------------------------

PMSPL Major Code: 0x00C6 Minor Code: 263 (0x0107)

Description: SplFSReturnCPAct Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplFSReturnCPAct
Minor Code: 263 (0x0107)
Trace Groups: FS
Trace Types: PRE
Traced Parameters: uHandle=%w
PMSPL Major Code: 0X00C6 Minor Code: 264 (0X0108)

Description
AttachPort Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.AttachPort

Minor Code
264 (0X0108)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
pPort=%F, PortName=%s

PMSPL Major Code: 0X00C6 Minor Code: 265 (0X0109)

Description
DetachPort Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.DetachPort

Minor Code
265 (0X0109)

Trace Groups
FS

Trace Types
PRE

Traced Parameters
pPort=%F, PortName=%s

PMSPL Major Code: 0X00C6 Minor Code: 304 (0X0130)

Description
PrintDestControl Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestControl
Minor Code  304 (0X0130)
Trace Groups  PRINTX
Trace Types  PRE
Traced Parameters
  pszDest=%s, Control=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 305 (0X0131)

Description  PrintDestGetInfo Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMSPL.PrintDestGetInfo
Minor Code  305 (0X0131)
Trace Groups  PRINTX
Trace Types  PRE
Traced Parameters
  pszName=%s, level=%F, cbBuf=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 306 (0X0132)

Description  PrintDestEnum Pre-Invocation
Tracepoint  Public symbol defined dynamic tracepoint: PMSPL.PrintDestEnum
Minor Code  306 (0X0132)
Trace Groups  PRINTX
Trace Types  PRE
Traced Parameters
  level=%F, cbBuf=%F
PMSPL Major Code: 0X00C6 Minor Code: 307 (0X0133)

Description
PrintDestAdd Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestAdd

Minor Code
307 (0X0133)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters
level%F, Buf=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 308 (0X0134)

Description
PrintDestSetInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestSetInfo

Minor Code
308 (0X0134)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters
pszPrinter=%s, level=%F, cbBuf=%F, parmnum=%F
Buf=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 309 (0X0135)

Description
PrintDestDel Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestDel

**Minor Code**
309 (0X0135)

**Trace Groups**
PRINTX

**Trace Types**
PRE

**Traced Parameters**
pszPrinter=%s, level=%F

PMSPL Major Code: 0X00C6 Minor Code: 320 (0X0140)

**Description**
PrintJobContinue Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintJobContinue

**Minor Code**
320 (0X0140)

**Trace Groups**
PRINTX

**Trace Types**
PRE

**Traced Parameters**
Job ID=%F, User Name=%s

PMSPL Major Code: 0X00C6 Minor Code: 321 (0X0141)

**Description**
PrintJobPause Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintJobPause

**Minor Code**
321 (0X0141)

**Trace Groups**
PRINTX

**Trace Types**
PRE

**Traced Parameters**
PMSPL Major Code: 0X00C6 Minor Code: 322 (0X0142)

Description: PrintJobDel Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintJobDel
Minor Code: 322 (0X0142)
Trace Groups: PRINTX
Trace Types: PRE
Traced Parameters:

Job ID=%F, User Name=%s

PMSPL Major Code: 0X00C6 Minor Code: 323 (0X0143)

Description: PrintJobSchedule Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintJobSchedule
Minor Code: 323 (0X0143)
Trace Groups: PRINTX
Trace Types: PRE
Traced Parameters:

Job ID=%w

PMSPL Major Code: 0X00C6 Minor Code: 324 (0X0144)

Description: PrintJobAdd Pre-Invocation
Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintJobAdd

Minor Code

324 (0X0144)

Trace Groups

PRINTX

Trace Types

PRE

Traced Parameters

level=%w, QueueName=%s

PMSPL Major Code: 0X00C6 Minor Code: 325 (0X0145)

Description

PrintJobGetInfo Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintJobGetInfo

Minor Code

325 (0X0145)

Trace Groups

PRINTX

Trace Types

PRE

Traced Parameters

Job ID=%F, level=%F, cbBuf=%F

PMSPL Major Code: 0X00C6 Minor Code: 326 (0X0146)

Description

PrintJobSetInfo Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintJobSetInfo

Minor Code

326 (0X0146)

Trace Groups

PRINTX

Trace Types

PRE

Traced Parameters
PMSPL Major Code: 0X00C6 Minor Code: 327 (0X0147)

Description: PrintJobEnum Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintJobEnum
Minor Code: 327 (0X0147)
Trace Groups: PRINTX
Trace Types: PRE
Traced Parameters: level=%F, cbBuf=%F, Qname=%s

PMSPL Major Code: 0X00C6 Minor Code: 336 (0X0150)

Description: PrintQPause Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQPause
Minor Code: 336 (0X0150)
Trace Groups: PRINTX
Trace Types: PRE
Traced Parameters: Qname=%s

PMSPL Major Code: 0X00C6 Minor Code: 337 (0X0151)

Description: PrintQPurge Pre-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQPurge

Minor Code: 337 (0X0151)

Trace Groups: PRINTX

Trace Types: PRE

Traced Parameters:
Qname=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 338 (0X0152)

Description: PrintQContinue Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQContinue

Minor Code: 338 (0X0152)

Trace Groups: PRINTX

Trace Types: PRE

Traced Parameters:
Qname=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 339 (0X0153)

Description: PrintQAdd Pre-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQAdd

Minor Code: 339 (0X0153)

Trace Groups: PRINTX

Trace Types: PRE

Traced Parameters: 
PMSPL Major Code: 0X00C6 Minor Code: 340 (0X0154)

Description
PrintQDel Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQDel

Minor Code
340 (0X0154)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters
Qname=%s, level=%F

PMSPL Major Code: 0X00C6 Minor Code: 341 (0X0155)

Description
PrintQGetInfo Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQGetInfo

Minor Code
341 (0X0155)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters
Qname=%s, level=%F, cbBuf=%F

PMSPL Major Code: 0X00C6 Minor Code: 342 (0X0156)

Description
PrintQSetInfo Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintQSetInfo

**Minor Code**
342 (0X0156)

**Trace Groups**
PRINTX

**Trace Types**
PRE

**Traced Parameters**

Qname=%s, level=%F, cbBuf=%F, parmnum=%F

PMSPL Major Code: 0X00C6 Minor Code: 343 (0X0157)

**Description**
PrintQEnum Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintQEnum

**Minor Code**
343 (0X0157)

**Trace Groups**
PRINTX

**Trace Types**
PRE

**Traced Parameters**

level=%F, cbBuf=%F

PMSPL Major Code: 0X00C6 Minor Code: 344 (0X0158)

**Description**
PrintDriverEnum Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintDriverEnum

**Minor Code**
344 (0X0158)

**Trace Groups**
PRINTX

**Trace Types**
PRE
Traced Parameters

level=%F, cbBuf=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 345 (0X0159)

Description
PrintQProcessorEnum Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQProcessorEnum

Minor Code
345 (0X0159)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters

level=%F, cbBuf=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 346 (0X015A)

Description
PrintPortEnum Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintPortEnum

Minor Code
346 (0X015A)

Trace Groups
PRINTX

Trace Types
PRE

Traced Parameters

level=%F, cbBuf=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 368 (0X0170)
Description SplWarning Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMSPL.SplWarning
Minor Code 368 (0X0170)
Trace Groups ERROR
Trace Types PRE
Traced Parameters Function=%s, ReturnCode=%F, Other=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 368 (0X0170)

Description SplRWarning Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMSPL.SplRWarning
Minor Code 368 (0X0170)
Trace Groups ERROR
Trace Types PRE
Traced Parameters Return Address=%F, rc=%F, value=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 369 (0X0171)

Description SplError Pre-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMSPL.SplError
Minor Code 369 (0X0171)
Trace Groups ERROR
Trace Types PRE
Traced Parameters

Function=%s, ReturnCode=%F, Other=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 369 (0X0171)

Description
SplRError Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplRError

Minor Code
369 (0X0171)

Trace Groups
ERROR

Trace Types
PRE

Traced Parameters

Return Address=%F, rc=%F, value=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 370 (0X0172)

Description
SplPanic Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplPanic

Minor Code
370 (0X0172)

Trace Groups
ERROR

Trace Types
PRE

Traced Parameters

Function=%s, ReturnCode=%F, Other=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 370 (0X0172)
### SplRPanic Pre-Invocation

**Description:** SplRPanic Pre-Invocation

**Tracepoint:** Public symbol defined dynamic tracepoint: PMSPL.SplRPanic

**Minor Code:** 370 (0X0172)

**Trace Groups:** ERROR

**Trace Types:** PRE

**Traced Parameters:**

- Return Address=%F, rc=%F, value=%F

---

### PMSPL Major Code: 0X00C6 Minor Code: 371 (0X0173)

**Description:** SplErrNotRunning Pre-Invocation

**Tracepoint:** Public symbol defined dynamic tracepoint: PMSPL.SplErrNotRunning

**Minor Code:** 371 (0X0173)

**Trace Groups:** ERROR

**Trace Types:** PRE

**Traced Parameters:** No parameters traced.

---

### PMSPL Major Code: 0X00C6 Minor Code: 372 (0X0174)

**Description:** SplErrNoMemory Pre-Invocation

**Tracepoint:** Public symbol defined dynamic tracepoint: PMSPL.SplErrNoMemory

**Minor Code:** 372 (0X0174)

**Trace Groups:** ERROR

**Trace Types:** PRE

**Traced Parameters:**
PMSPL Major Code: 0X00C6 Minor Code: 373 (0X0175)

Description
SplLogWarning Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplLogWarning

Minor Code
373 (0X0175)

Trace Groups
ERROR

Trace Types
PRE

Traced Parameters
PMError=%F

PMSPL Major Code: 0X00C6 Minor Code: 374 (0X0176)

Description
SplLogError Pre-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplLogError

Minor Code
374 (0X0176)

Trace Groups
ERROR

Trace Types
PRE

Traced Parameters
PMError=%F

PMSPL Major Code: 0X00C6 Minor Code: 384 (0X0180)

Description
PostRefreshMsg Pre-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PostRefreshMsg

**Minor Code**
384 (0x180)

**Trace Groups**
REFRESH

**Trace Types**
PRE

**Traced Parameters**
Type=%F, JobID=%F, Queue=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32770 (0X8002)

**Description**
SplControlDevice Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplControlDevice

**Minor Code**
32770 (0X8002)

**Trace Groups**
DOS

**Trace Types**
POST

**Traced Parameters**
rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32771 (0X8003)

**Description**
SplQueryDevice Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.SplQueryDevice

**Minor Code**
32771 (0X8003)

**Trace Groups**
DOS

**Trace Types**
POST
Traced Parameters

rc=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32772 (0X8004)

Description
SplEnumDevice Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplEnumDevice

Minor Code
32772 (0X8004)

Trace Groups
DOS

Trace Types
POST

Traced Parameters

rc=%F, cbNeeded=%F, cReturned=%F, cTotal=%F
pBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32773 (0X8005)

Description
SplCreateDevice Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.SplCreateDevice

Minor Code
32773 (0X8005)

Trace Groups
DOS

Trace Types
POST

Traced Parameters

rc=%F, pBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32774 (0X8006)
SplSetDevice Post-Invocation

Description: SplSetDevice Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplSetDevice
Minor Code: 32774 (0X8006)
Trace Groups: DOS
Trace Types: POST
Traced Parameters: rc=%F, pBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32775 (0X8007)

Description: SplDeleteDevice Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplDeleteDevice
Minor Code: 32775 (0X8007)
Trace Groups: DOS
Trace Types: POST
Traced Parameters: rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32776 (0X8008)

Description: SplReleaseJob Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplReleaseJob
Minor Code: 32776 (0X8008)
Trace Groups: DOS
Trace Types      POST

Traced Parameters

rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32777 (0X8009)

Description      SplHoldJob Post-Invocation
Tracepoint       Public symbol defined dynamic tracepoint: PMSPL.SplHoldJob
Minor Code       32777 (0X8009)
Trace Groups     DOS
Trace Types      POST
Traced Parameters

rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32778 (0X800A)

Description      SplDeleteJob Post-Invocation
Tracepoint       Public symbol defined dynamic tracepoint: PMSPL.SplDeleteJob
Minor Code       32778 (0X800A)
Trace Groups     DOS
Trace Types      POST
Traced Parameters

rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32779 (0X800B)
<table>
<thead>
<tr>
<th>Description</th>
<th>DosPrintJobSchedule Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobSchedule</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32779 (0X800B)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>DosPrintJobAdd Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32780 (0X800C)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F, pulJob=%w, pszFileName=%s</td>
</tr>
</tbody>
</table>

--------------------------------------------

<table>
<thead>
<tr>
<th>Description</th>
<th>DosPrintJobAdd2 Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd2</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32781 (0X800D)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
</tbody>
</table>
Trace Types: POST

Traced Parameters:
rc=%F, pulJob=%w, pszFileName=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32782 (0X800E)

Description: SplQueryJob Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplQueryJob
Minor Code: 32782 (0X800E)
Trace Groups: DOS
Trace Types: POST
Traced Parameters:
rc=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32783 (0X800F)

Description: SplSetJob Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplSetJob
Minor Code: 32783 (0X800F)
Trace Groups: DOS
Trace Types: POST
Traced Parameters:
rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32784 (0X8010)
<table>
<thead>
<tr>
<th>Description</th>
<th>SplEnumJob Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplEnumJob</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32784 (0X8010)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32785 (0X8011)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplCreateQueue Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplCreateQueue</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32785 (0X8011)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32786 (0X8012)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplHoldQueue Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplHoldQueue</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32786 (0X8012)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
</tbody>
</table>
Trace Types: POST
Traced Parameters:
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32787 (0X8013)

Description: SplReleaseQueue Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplReleaseQueue
Minor Code: 32787 (0X8013)
Trace Groups: DOS
Trace Types: POST
Traced Parameters:
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32788 (0X8014)

Description: SplDeleteQueue Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplDeleteQueue
Minor Code: 32788 (0X8014)
Trace Groups: DOS
Trace Types: POST
Traced Parameters:
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32789 (0X8015)
SplPurgeQueue Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.SplPurgeQueue

Minor Code
32789 (0X8015)

Trace Groups
DOS

Trace Types
POST

Traced Parameters
rc=%F

--------------------------------------------

SplQueryQueue Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.SplQueryQueue

Minor Code
32790 (0X8016)

Trace Groups
DOS

Trace Types
POST

Traced Parameters
rc=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------

SplSetQueue Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.SplSetQueue

Minor Code
32791 (0X8017)

Trace Groups
DOS
trace types

POST

traced parameters

rc=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32792 (0X8018)

description

SplEnumQueue Post-Invocation

tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueue

minor code

32792 (0X8018)

trace groups

DOS

trace types

POST

traced parameters

rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pbBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32793 (0X8019)

description

SplEnumDriver Post-Invocation

tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplEnumDriver

minor code

32793 (0X8019)

trace groups

DOS

trace types

POST

traced parameters

rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pbBuf=%r%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32794 (0X801A)
**Description**  SplEnumQueueProcessor Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueueProcessor

**Minor Code**  32794 (0X801A)

**Trace Groups**  DOS

**Trace Types**  POST

**Traced Parameters**  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------

**Description**  SplEnumPort Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: PMSPL.SplEnumPort

**Minor Code**  32795 (0X801B)

**Trace Groups**  DOS

**Trace Types**  POST

**Traced Parameters**  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------

**Description**  DosPrintJobGetId Post-Invocation

**Tracepoint**  Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobGetId

**Minor Code**  32796 (0X801C)

**Trace Groups**  DOS

**Trace Types**  POST

**Traced Parameters**  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

--------------------------------------------
<table>
<thead>
<tr>
<th><strong>Trace Types</strong></th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>rc=%F, JobID =%F pInfo=%r%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32797 (0X801D)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Spl32PrmSpool Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32PrmSpool</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32797 (0X801D)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>rc=%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32798 (0X801E)

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>SplQueryDriver Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracepoint</strong></td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplQueryDriver</td>
</tr>
<tr>
<td><strong>Minor Code</strong></td>
<td>32798 (0X801E)</td>
</tr>
<tr>
<td><strong>Trace Groups</strong></td>
<td>DOS</td>
</tr>
<tr>
<td><strong>Trace Types</strong></td>
<td>POST</td>
</tr>
<tr>
<td><strong>Traced Parameters</strong></td>
<td>rc=%F, cbNeeded=%F, pBuf=%r%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32799 (0X801F)
<table>
<thead>
<tr>
<th>Description</th>
<th>SplSetDriver Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplSetDriver</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32799 (0X801F)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F, pBuf=%r%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32800 (0X8020)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplCopyJob Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplCopyJob</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32800 (0X8020)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F, NewJobID=%F</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 32801 (0X8021)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplQueryJobFile Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplQueryJobFile</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32801 (0X8021)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>DOS</td>
</tr>
</tbody>
</table>

**Trace Types**

POST

**Traced Parameters**

rc=%F, FileName=%s

__________________________________________

**PMSPL Major Code: 0X00C6 Minor Code: 32802 (0X8022)**

**Description**

SplEnumPrinter Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplEnumPrinter

**Minor Code**

32802 (0X8022)

**Trace Groups**

DOS

**Trace Types**

POST

**Traced Parameters**

rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%s

__________________________________________

**PMSPL Major Code: 0X00C6 Minor Code: 32817 (0X8031)**

**Description**

Spl32QmOpen Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmOpen

**Minor Code**

32817 (0X8031)

**Trace Groups**

SPLQM

**Trace Types**

POST

**Traced Parameters**

Spool File Handle=%F, Possible ErrorCode=%F

__________________________________________

**PMSPL Major Code: 0X00C6 Minor Code: 32818 (0X8032)**
Spl32QmStartDoc Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmStartDoc

Minor Code
32818 (0X8032)

Trace Groups
SPLQM

Trace Types
POST

Traced Parameters

fSuccess=%F

--------------------------------------------------

Spl32QmWrite Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWrite

Minor Code
32819 (0X8033)

Trace Groups
SPLQM

Trace Types
POST

Traced Parameters

fSuccess=%F

--------------------------------------------------

Spl32QmWriteFile Post-Invocation

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWriteFile

Minor Code
32820 (0X8034)

Trace Groups
SPLQM
Trace Types
POST

Traced Parameters
fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32821 (0X8035)

Description
Spl32QmEndDoc Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmEndDoc

Minor Code
32821 (0X8035)

Trace Groups
SPLQM

Trace Types
POST

Traced Parameters
Job ID=%F

PMSPL Major Code: 0X00C6 Minor Code: 32822 (0X8036)

Description
Spl32QmAbortDoc Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbortDoc

Minor Code
32822 (0X8036)

Trace Groups
SPLQM

Trace Types
POST

Traced Parameters
fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32823 (0X8037)
<table>
<thead>
<tr>
<th>Description</th>
<th>Spl32QmClose Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32QmClose</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32823 (0X8037)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SPLQM</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>fSuccess=%F</td>
</tr>
</tbody>
</table>

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32824 (0X8038)

<table>
<thead>
<tr>
<th>Description</th>
<th>Spl32QmAbort Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbort</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32824 (0X8038)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SPLQM</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>fSuccess=%F</td>
</tr>
</tbody>
</table>

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 32825 (0X8039)

<table>
<thead>
<tr>
<th>Description</th>
<th>Spl32QmQueryPinfo Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32QmQueryPinfo</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32825 (0X8039)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>SPLQM</td>
</tr>
</tbody>
</table>
Trace Types

POST

Traced Parameters

Job ID=%w

PMSPL Major Code: 0X00C6 Minor Code: 32826 (0X803A)

Description

Spl32QmSetStatus Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetStatus

Minor Code

32826 (0X803A)

Trace Groups

SPLQM

Trace Types

POST

Traced Parameters

fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32827 (0X803B)

Description

Spl32QmSetup Post-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetup

Minor Code

32827 (0X803B)

Trace Groups

SPLQM

Trace Types

POST

Traced Parameters

rc=%F Type=%F DopData=%F %F %F %F %F %F LogAddr=%s DriverName=%s DataType=%s pdriv->cb=%F pdriv->lVersion=%F pdriv->szDeviceName=%s
### PMSPL Major Code: 0X00C6 Minor Code: 32849 (0X8051)

<table>
<thead>
<tr>
<th>Description</th>
<th>Spl32MessageBox Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Spl32MessageBox</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32849 (0X8051)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRT</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 32850 (0X8052)

<table>
<thead>
<tr>
<th>Description</th>
<th>Prt32Open Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Prt32Open</td>
</tr>
<tr>
<td>Minor Code</td>
<td>32850 (0X8052)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRT</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F, hFile=%F, ActionTaken=%F, Dos Filehandle=%F</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 32851 (0X8053)

<table>
<thead>
<tr>
<th>Description</th>
<th>Prt32Write Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.Prt32Write</td>
</tr>
</tbody>
</table>
Minor Code: 32851 (0X8053)
Trace Groups: PRT
Trace Types: POST
Traced Parameters: RC=%F, cbWritten=%F

PMSPL Major Code: 0X00C6 Minor Code: 32852 (0X8054)

Description: Prt32DevIOCtl Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Prt32DevIOCtl
Minor Code: 32852 (0X8054)
Trace Groups: PRT
Trace Types: POST
Traced Parameters: RC=%F

PMSPL Major Code: 0X00C6 Minor Code: 32853 (0X8055)

Description: Prt32Close Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Prt32Close
Minor Code: 32853 (0X8055)
Trace Groups: PRT
Trace Types: POST
Traced Parameters: RC=%F
PMSPL Major Code: 0X00C6 Minor Code: 32854 (0X8056)

Description
Prt32Abort Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.Prt32Abort

Minor Code
32854 (0X8056)

Trace Groups
PRT

Trace Types
POST

Traced Parameters
RC=%F

PMSPL Major Code: 0X00C6 Minor Code: 32855 (0X8057)

Description
PrtNewPage Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrtNewPage

Minor Code
32855 (0X8057)

Trace Groups
PRT

Trace Types
POST

Traced Parameters
RC=%F

PMSPL Major Code: 0X00C6 Minor Code: 32856 (0X8058)

Description
PrtResetAbort Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrtResetAbort
Minor Code: 32856 (0X8058)
Trace Groups: PRT
Trace Types: POST
Traced Parameters:
RC=%F

PMSPL Major Code: 0X00C6 Minor Code: 32857 (0X8059)

Description: PrtAbortDoc Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrtAbortDoc
Minor Code: 32857 (0X8059)
Trace Groups: PRT
Trace Types: POST
Traced Parameters:
RC=%F

PMSPL Major Code: 0X00C6 Minor Code: 32881 (0X8071)

Description: Spl32StdOpen Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdOpen
Minor Code: 32881 (0X8071)
Trace Groups: STD
Trace Types: POST
Traced Parameters:
fSuccess=%F
PMSPL Major Code: 0X00C6 Minor Code: 32882 (0X8072)

Description: Spl32StdClose Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdClose
Minor Code: 32882 (0X8072)
Trace Groups: STD
Trace Types: POST
Traced Parameters:

fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32883 (0X8073)

Description: Spl32StdStart Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdStart
Minor Code: 32883 (0X8073)
Trace Groups: STD
Trace Types: POST
Traced Parameters:

fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32884 (0X8074)

Description: Spl32StdStop Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdStop
Minor Code: 32884 (0X8074)

Trace Groups: STD

Trace Types: POST

Traced Parameters: pBuffer(HSDT)=%F

PMSPL Major Code: 0X00C6 Minor Code: 32885 (0X8075)

Description: Spl32StdQueryLength Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdQueryLength

Minor Code: 32885 (0X8075)

Trace Groups: STD

Trace Types: POST

Traced Parameters: Length=%F

PMSPL Major Code: 0X00C6 Minor Code: 32886 (0X8076)

Description: Spl32StdGetBits Post-Invocation

Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.Spl32StdGetBits

Minor Code: 32886 (0X8076)

Trace Groups: STD

Trace Types: POST

Traced Parameters: fSuccess=%F
PMSPL Major Code: 0X00C6 Minor Code: 32887 (0X8077)

**Description**  
Spl32StdDelete Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.Spl32StdDelete

**Minor Code**  
32887 (0X8077)

**Trace Groups**  
STD

**Trace Types**  
POST

**Traced Parameters**

fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 33024 (0X8100)

**Description**  
SplFSOpen Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplFSOpen

**Minor Code**  
33024 (0X8100)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**

rc=%w, JobId=%w

PMSPL Major Code: 0X00C6 Minor Code: 33025 (0X8101)

**Description**  
SplFSFirstWrite Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplFSFirstWrite
<table>
<thead>
<tr>
<th>Minor Code</th>
<th>33025 (0X8101)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Error=%w, CP Activation length=%w</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 33026 (0X8102)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSWriteFail Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSWriteFail</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>33026 (0X8102)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>Error=%w, Response=%w</td>
</tr>
</tbody>
</table>

PMSPL Major Code: 0X00C6 Minor Code: 33027 (0X8103)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSClose Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSClose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minor Code</th>
<th>33027 (0X8103)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%w</td>
</tr>
</tbody>
</table>
### PMSPL Major Code: 0X00C6 Minor Code: 33028 (0X8104)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSSetTitle Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSSetTitle</td>
</tr>
<tr>
<td>Minor Code</td>
<td>33028 (0X8104)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%w</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 33029 (0X8105)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSActCP Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSActCP</td>
</tr>
<tr>
<td>Minor Code</td>
<td>33029 (0X8105)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>FS</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%w</td>
</tr>
</tbody>
</table>

### PMSPL Major Code: 0X00C6 Minor Code: 33030 (0X8106)

<table>
<thead>
<tr>
<th>Description</th>
<th>SplFSVerifyCP Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.SplFSVerifyCP</td>
</tr>
</tbody>
</table>

Minor Code: 33030 (0X8106)
Trace Groups: FS
Trace Types: POST
Traced Parameters:
DosPFSVerifyFont rc=%w

PMSPL Major Code: 0X00C6 Minor Code: 33031 (0X8107)
Description: SplFSReturnCPAct Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.SplFSReturnCPAct
Minor Code: 33031 (0X8107)
Trace Groups: FS
Trace Types: POST
Traced Parameters:
Error=%w, CP Activation length=%w

PMSPL Major Code: 0X00C6 Minor Code: 33032 (0X8108)
Description: AttachPort Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.AttachPort
Minor Code: 33032 (0X8108)
Trace Groups: FS
Trace Types: POST
Traced Parameters:
rc=%w
PMSPL Major Code: 0X00C6 Minor Code: 33033 (0X8109)

**Description**
- DetachPort Post-Invocation

**Tracepoint**
- Public symbol defined dynamic tracepoint: PMSPL.DetachPort

**Minor Code**
- 33033 (0X8109)

**Trace Groups**
- FS

**Trace Types**
- POST

**Traced Parameters**
- rc=%w

PMSPL Major Code: 0X00C6 Minor Code: 33034 (0X810A)

**Description**
- GetNextId Post-Invocation

**Tracepoint**
- Public symbol defined dynamic tracepoint: PMSPL.GetNextId

**Minor Code**
- 33034 (0X810A)

**Trace Groups**
- PRINTX

**Trace Types**
- POST

**Traced Parameters**
- Returned Job ID =%w

PMSPL Major Code: 0X00C6 Minor Code: 33072 (0X8130)

**Description**
- PrintDestControl Post-Invocation

**Tracepoint**
- Public symbol defined dynamic tracepoint: PMSPL.PrintDestControl
PMSPL Major Code: 0X00C6 Minor Code: 33072 (0X8130)

Description
PrintDestGetInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestGetInfo

Minor Code
33072 (0X8130)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33073 (0X8131)

Description
PrintDestGetInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestGetInfo

Minor Code
33073 (0X8131)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F, *pcbNeeded=%F

PMSPL Major Code: 0X00C6 Minor Code: 33074 (0X8132)

Description
PrintDestEnum Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestEnum

Minor Code
33074 (0X8132)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F
PMSPL Major Code: 0X00C6 Minor Code: 33075 (0X8133)

Description
PrintDestAdd Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestAdd

Minor Code
33075 (0X8133)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33076 (0X8134)

Description
PrintDestSetInfo Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestSetInfo

Minor Code
33076 (0X8134)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33077 (0X8135)

Description
PrintDestDel Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintDestDel
PMSPL Major Code: 0X00C6 Minor Code: 33088 (0X8140)

Description: PrintJobContinue Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintJobContinue

PMSPL Major Code: 0X00C6 Minor Code: 33089 (0X8141)

Description: PrintJobPause Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintJobPause
# PMSPL Major Code: 0X00C6 Minor Code: 33090 (0X8142)

<table>
<thead>
<tr>
<th>Description</th>
<th>PrintJobDel Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.PrintJobDel</td>
</tr>
<tr>
<td>Minor Code</td>
<td>33090 (0X8142)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRINTX</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%F</td>
</tr>
</tbody>
</table>

# PMSPL Major Code: 0X00C6 Minor Code: 33091 (0X8143)

<table>
<thead>
<tr>
<th>Description</th>
<th>PrintJobSchedule Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.PrintJobSchedule</td>
</tr>
<tr>
<td>Minor Code</td>
<td>33091 (0X8143)</td>
</tr>
<tr>
<td>Trace Groups</td>
<td>PRINTX</td>
</tr>
<tr>
<td>Trace Types</td>
<td>POST</td>
</tr>
<tr>
<td>Traced Parameters</td>
<td>rc=%w</td>
</tr>
</tbody>
</table>

# PMSPL Major Code: 0X00C6 Minor Code: 33092 (0X8144)

<table>
<thead>
<tr>
<th>Description</th>
<th>PrintJobAdd Post-Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracepoint</td>
<td>Public symbol defined dynamic tracepoint: PMSPL.PrintJobAdd</td>
</tr>
</tbody>
</table>
Minor Code 33092 (0X8144)
Trace Groups PRINTX
Trace Types POST
Traced Parameters

rc=%w, JobID=%w, Filename=%s

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 33093 (0X8145)

Description PrintJobGetInfo Post-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMSPL.PrintJobGetInfo
Minor Code 33093 (0X8145)
Trace Groups PRINTX
Trace Types POST
Traced Parameters

rc=%F, cbNeeded=%F

--------------------------------------------

PMSPL Major Code: 0X00C6 Minor Code: 33094 (0X8146)

Description PrintJobSetInfo Post-Invocation
Tracepoint Public symbol defined dynamic tracepoint: PMSPL.PrintJobSetInfo
Minor Code 33094 (0X8146)
Trace Groups PRINTX
Trace Types POST
Traced Parameters

rc=%F
PMSPL Major Code: 0X00C6 Minor Code: 33095 (0X8147)

**Description**
PrintJobEnum Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintJobEnum

**Minor Code**
33095 (0X8147)

**Trace Groups**
PRINTX

**Trace Types**
POST

**Traced Parameters**
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

PMSPL Major Code: 0X00C6 Minor Code: 33104 (0X8150)

**Description**
PrintQPause Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintQPause

**Minor Code**
33104 (0X8150)

**Trace Groups**
PRINTX

**Trace Types**
POST

**Traced Parameters**
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33105 (0X8151)

**Description**
PrintQPurge Post-Invocation

**Tracepoint**
Public symbol defined dynamic tracepoint: PMSPL.PrintQPurge
PMSPL Major Code: 0X00C6 Minor Code: 33105 (0X8151)

Description
PrintQContinue Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQContinue

PMSPL Major Code: 0X00C6 Minor Code: 33106 (0X8152)

Description
PrintQContinue Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQContinue

PMSPL Major Code: 0X00C6 Minor Code: 33107 (0X8153)

Description
PrintQAdd Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQAdd
PMSPL Major Code: 0X00C6 Minor Code: 33108 (0X8154)

Description: PrintQDel Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQDel
Minor Code: 33108 (0X8154)
Trace Groups: PRINTX
Trace Types: POST
Traced Parameters: rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33109 (0X8155)

Description: PrintQGetInfo Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQGetInfo
Minor Code: 33109 (0X8155)
Trace Groups: PRINTX
Trace Types: POST
Traced Parameters: rc=%F, cbNeeded=%F

PMSPL Major Code: 0X00C6 Minor Code: 33110 (0X8156)

Description: PrintQSetInfo Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQSetInfo
Minor Code: 33110 (0X8156)
Trace Groups: PRINTX
Trace Types: POST
Traced Parameters: rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 33111 (0X8157)

Description: PrintQEnum Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintQEnum
Minor Code: 33111 (0X8157)
Trace Groups: PRINTX
Trace Types: POST
Traced Parameters: rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

PMSPL Major Code: 0X00C6 Minor Code: 33112 (0X8158)

Description: PrintDriverEnum Post-Invocation
Tracepoint: Public symbol defined dynamic tracepoint: PMSPL.PrintDriverEnum
Minor Code: 33112 (0X8158)
Trace Groups: PRINTX
Trace Types: POST
Traced Parameters: rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F
PMSPL Major Code: 0X00C6 Minor Code: 33113 (0X8159)

Description
PrintQProcessorEnum Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintQProcessorEnum

Minor Code
33113 (0X8159)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

PMSPL Major Code: 0X00C6 Minor Code: 33114 (0X815A)

Description
PrintPortEnum Post-Invocation

Tracepoint
Public symbol defined dynamic tracepoint: PMSPL.PrintPortEnum

Minor Code
33114 (0X815A)

Trace Groups
PRINTX

Trace Types
POST

Traced Parameters
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

CONFIG.SYS RAS Statements

OS/2 provides a number of CONFIG.SYS commands and settings specifically for RAS purposes. Some of these are described in the OS/2 Command Reference, and will not be discussed in detail here. A number of commands were introduced or enhanced in APAR PJ12256, which is applicable to OS/2 2.11. All the commands described here are available with OS/2 Warp 3.0.

The following CONFIG.SYS commands comprise the RAS set. Those not completely documented in the OS/2 Command Reference are now discussed in detail.
• AUTOFAIL (see the OS/2 Command Reference).
• DUMPPROCESS
• REIPL
• RASKDATA
• SCKILLFEATUREENABLED
• SHAPIEXCEPTIONHANDLER
• SHELLEXCEPTIONHANDLER
• SUPPRESSPOPUPS
• TRACE (see the OS/2 Command Reference). Also see the System Trace Facility - User Guide chapter in this book.
• TRACEDUMP (see the OS/2 Command Reference).
• TRAPDUMP

AUTOFAIL

AUTOFAIL modifies the processing of media errors. See the OS/2 Command Reference.

DUMPPROCESS

This command allows the user to activate the process dump facility. When active, any ring 3 (application) process that traps will result in a memory dump being written to a unique dump file. The dump file takes a name of the form \texttt{PDUMP:nnn} where \texttt{nnn} is an index that is incremented each time a new process dump is created.

The contents of the dump comprise unformatted system and user storage that relates to the trapping process. Included in this are:

• PTDA
• TCB &TSD
• Registers
• Arena records
• MTE and SMTEs
• LDT
• ring 0 stack
• ring 3 stack

\textbf{Syntax}

\texttt{DUMPPROCESS=x}

\textbf{Parameters}
This specifies the drive letter (excluding the colon) to which process dump data sets will be written. These takes the name `PDUMP.nnn` and reside in the root directory of the drive specified. The name and directory cannot be overridden by the user.

**Note:**


---

**REIPL**

This command allows the user to automate the re-booting (re-IPLing) of the system following an IPE.

**Syntax.**

```
REIPL=ON | OFF
```

**Parameters**

**ON**

This specifies the system it to be automatically re-booted following an IPE.

**OFF**

This specified that the system is not to be automatically re-booted following an IPE. The system will remain hung until manually restarted.

**Notes:**

- **REIPL** only applied to pre-WARP systems if APAR PJ12258 has been applied.
- **REIPL** has no effect when TRAPDUMP=ON/R0 is specified. Whether the system is re-booted following a stand-alone dump is governed by the `OS2DUMP` module. If the the dump is to hard-disk then automatic re-boot occurs, otherwise not.

---

**RASKDATA**

The RASKDATA statement directs the kernel to retain various system control blocks in memory for debugging purposes.

**Syntax.**

```
RASKDATA=[OTE] [,LOCKS | NOLOCKS]
```

**Parameters**

**OTE**

causes the **OTE**, **STE** and **SMTE** loader structures to be retained in resident memory for use by the Dump Formatter .LM command.

The default behaviour is for **OTE** and **SMTE** structure to be allocated from resident memory, which can causes these to become paged out and thus not present in a system dump.

**LOCKS**

causes long and short term lock records to be retained for use by the Dump Formatter .MK command.
The default behaviour under the RETAIL kernel is not to retain lock records.
The default behaviour under the HSTRINGT and ALLSTRINGT kernels is to retain lock records.

LOCKS and NOLOCKS are mutually exclusive options.

NOLOCKS
causes long and short term lock records not to be retained. If this option is specified then the Dump Formatter .MK command will not function.

The default behaviour under the RETAIL kernel is not to retain lock records.
The default behaviour under the HSTRINGT and ALLSTRINGT kernels is to retain lock records.

Remarks
Use of RASKDATA will impact resident memory requirements may force greater paging activity if insufficient memory is available.
The OTE option was introduced with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.
The LOCKS and NOLOCKS options were introduced with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10.
All options are available with OS/2 Warp E-Serv

--------------------------------------------

SCKILLFEATUREENABLED

This command enables the Kill Feature of the Warp Centre introduced with OS/2 Warp V4.0.

Syntax.

SET SCKILLFEATUREENABLED= YES | NO

Parameters

YES
Enables the kill Feature of the Warp Centre.

NO
Disables the kill Feature of the Warp Centre. This is the default setting.

Note:
The Kill Feature is invoked by clicking on the window list icon of the Warp Centre while holding down the Ctrl key. This causes a list of all executing processes to be displayed. Selecting a process from this list will cause it to be terminated, following a confirmation prompt.

--------------------------------------------

SHAPIEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the PMSHAPI.DLL module.

Syntax.

SET SHAPIEXCEPTIONHANDLER=ON | OFF

Parameters
ON
This is the default setting. The shell API DLL exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

OFF
The shell API DLL exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

Notes:
Exception handler registration only occurs during PMSHAPI.DLL initialisation. Therefore, a change to the specification of SHAPIEXCEPTIONHANDLER will require the system to be rebooted.

The Shell API DLL exception handler will attempt to clean up an application's PM resources.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analysed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. SHAPIEXCEPTIONHANDLER may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

SHAPIEXCEPTIONHANDLER may be used with TRAPDUMP to force a system dump at the point of failure.

Hangs in the shell during initialisation may be the result of a re-cursive trap. SHAPIEXCEPTIONHANDLER may be used to intercept this condition.

Since it if difficult to determine whether a potential shell problem involves PMSHELL.EXE or PMSHAPI.DLL then it is recommended to use SHAPIEXCEPTIONHANDLER with SHELEXCEPTIONHANDLER.

--------------------------------------------

SHELEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the PMSHELL.EXE module.

Syntax.
SET SHELEXCEPTIONHANDLER=ON|OFF

Parameters

ON
This is the default setting. The shell's exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

OFF
The shell's exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

Notes:
Exception handler registration only occurs during PMSHELL.EXE initialisation. Therefore, a change to the specification of SHELEXCEPTIONHANDLER will require the system to be rebooted.

The Shell's exception handler will attempt to clean up an application's PM resources. In addition if the application is the Desktop (or whatever is specified in RUNWORKPLACE), then it is restarted.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analysed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. SHELEXCEPTIONHANDLER may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

SHELEXCEPTIONHANDLER may be used with TRAPDUMP to force a system dump at the point of failure.

Hangs in the shell during initialisation may be the result of a re-cursive trap. SHELEXCEPTIONHANDLER may be used to
intercept this condition.

Since it is difficult to determine whether a potential shell problem involves PMSHELL.EXE or PMSHAPI.DLL then it is recommended to use SHELEXCEPTIONHANDLER with SHAPIEXCEPTIONHANDLER.

--------------------------------------------

SUPPRESSPOPUPS

This command allows the user to suppress the display of trap information pop-up messages and instead direct trap information to a log data set.

Syntax

SUPPRESSPOPUPS=x

Parameters

x

This specifies the drive letter (excluding the colon) to which the pop-up log data set will be written. The log takes the name POPUPLOG.OS2 and resides in the root directory of the drive specified. The name and directory cannot be overridden by the user.

From Fix Pack 29 of OS/2 Warp V3.0 and OS/2 Warp V4.0 GA, trap screen pop-ups are by default logged by default in POPUPLOG.OS2 in to root directory of the boot drive and not suppressed. x may specify 0 to disable automatic logging. In addition, the TRAPLOG command provides a command line interface to control trap screen logging and suppression independently.

--------------------------------------------

TRACE

TRACE specifies whether tracing of static trace events is to be active from system initialisation or not. See the OS/2 Command Reference for details. See also the System Trace Facility - User Guide chapter in this book.

--------------------------------------------

TRACEBUF

TRACEBUF specifies the size of the system trace buffer. See the OS/2 Command Reference for details.

--------------------------------------------

TRACEFMT

The TRACEFMT utility is used to extract and format the system trace from the either a saved trace buffer or the currently active trace buffer. See the OS/2 Command Reference for details.

--------------------------------------------

TRAPDUMP

Warning: Potential Data Loss
Misuse of this facility may cause loss of vital data. Please read carefully the complete description before use.

The TRAPDUMP command controls the stand-alone (system) dump facility of OS/2. It will enable initiation of a stand-alone dump at the instant a ring 3 trap occurs for which no exception handler has intervened.

Ring 0 traps may be also intercepted only on 2.11 systems to which APAR PJ12258 has been applied (kernel revision 6.624), or on OS/2 Warp.

Pre-Warp considerations: The dump process is performed by the hidden module OS2DUMP, which resides in the root directory of the boot drive. OS2DUMP as supplied with GA versions of OS/2 2.x dumps only to diskette. It may be replaced with a version supplied with OS/2 Problem Determination Package (OS2PDP) which will dump to a hard disk FAT partition that has volume label SADUMP or to diskette, depending upon TRAPDUMP command specification.

The GA 2.x version of OS2DUMP requires the first dump diskette be freshly prepared using the CREATEDD command and subsequent diskettes to be formatted. See the on-line OS/2 Command Reference for details of CREATEDD.

The OS/2 Problem Determination Package (OS2PDP) version of OS2DUMP only requires formatted diskettes, the use of CREATEDD being redundant.

When dumping to hard disk the dump partition must to be made known to TRAPDUMP. This is done by specifying an optional second parameter.

OS/2 Warp considerations: Under OS/2 Warp the CREATEDD command is unnecessary and is not distributed with the system. Ordinarily formatted diskettes may be used. Furthermore the enhanced version of OS2DUMP that allows dumping to a hard-disk FAT partition is standard. The partition volume label must be SADUMP.

Syntax.

TRAPDUMP=[ON|OFF|R0][,PD][,]X:

Parameters

ON
Specifies that the stand-alone dump process will be automatically initiated whenever an unrecoverable ring 3 trap occurs. For 2.11 systems with APAR PJ12258 or OS/2 Warp, any system IPE (including ring 0 traps) will also initiate a dump when ON is specified.

OFF
Specifies that the stand-alone dump process will not initiate automatically when an unrecoverable trap occurs. This is the default option. It does not prohibit the use of the Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequence or the use of DosForceSystemDump to force a system dump to be initiated.

R0
Specifies that only ring zero traps and IPEs will automatically initiate the stand-alone dump process. This option applies only to 2.11 systems with APAR PJ12258 or OS/2 Warp.

Note:
When an IPE occurs the dump is taken immediately on displaying the IPE trap screen. For the purposes of dump analysis the formatted registers from the IPE screen should be located from the video buffer, which may be viewed using the analyse option from PMDF.

PD
Specifies that a system level process dump will be attempted if the user uses either the Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequences. If for some reason the process dump does not compete, a second key sequence will initiate a system dump. On completion of either the system or process dump, the system is re-booted automatically.

X:
specifies the hard-disk FAT partition to which OS2DUMP will write a stand-alone dump. The partition letter must have the colon suffix.

Note:
The partition may be specified with either ON or OFF. When specified with OFF it will allow a stand-alone dump initiated by Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequences to be written to the dump partition.

Mountable media other than diskette drives are not detectable by OS2DUMP. The letter specifying the dump partition must be
calculated as if any such media were not present.

Only hard disk logical drives and primary partitions may be specified.

When dumping to a hard disk partition is selected the system is automatically re-booted on completion of the dump.

System Dump parameter may also be set dynamically from the command line by using the TRAPDUMP command.

Warnings:
The stand-alone dump process will erase all data on the dump media (disk partition or diskettes) before writing the dump.

Do not specify a disk partition or use diskettes that contain vital data.

--------------------------------------------

Miscellaneous RAS Command Utilities

OS/2 provides a number of command line utilities for use in problem determination and controlling system RAS settings dynamically. The commands described in this section have not been formally documented at all versions of OS/2 and so are described here.

The commands described are:

• TRAPDUMP
• TRAPLOG
• SYSDUMP

--------------------------------------------

TRAPDUMP

The TRAPDUMP command allows the conditions under which a trap will initiate a System Dump to be set dynamically.

Warning:
The initiation of a System Dump causes an immediate termination of the system without any shutdown. No file system shutdown is performed. The system behaves as if a fatal crash has occurred, thus under rare circumstances data can be lost.

Syntax:

TRAPDUMP [[ON] | [OFF] | [R0]] [x:] [/NOCHECK] [QUERY]

Parameters

ON

enables all application and system traps to initiate a System Dump.

OFF

disables automatic dump initiation.

R0

enables only Ring 0 traps to initiate a System Dump.

x:

specifies the Dump Partition.
disable the check for system level OS/2 Warp V4.0. This parameter is now obsolete, since the TRAPDUMP command is now available on all supported releases of OS/2.

shows the current settings for TRAPDUMP as a CONFIG.SYS statement and an equivalent command line command.

ON, OFF and R0 parameters are mutually exclusive.

The PD option of the CONFIG.SYS TRAPDUMP cannot be set dynamically.

TRAPDUMP was made available at fix pack 29 for OS/2 Warp V3.0 and base releases of later versions of OS/2.

TRAPLOG

The TRAPLOG command allows dynamic control of trap and exceptions popup message logging and display.

Syntax.

TRAPLOG [x: | NOLOG] [POPUPS | NOPOPUPS]

Parameters.

\( x: \)

specifies that trap information is to be logged in \( x:\)POPUPLOG.OS2, \( x: \) being any partition drive letter.

NOLOG

disables logging of trap information.

POPUPS

enables the trap information pop-up message (SYS317x).

NOPOPUPS

disables the trap information pop-up message.

If neither POPUPS nor NOPOPUPS is specified then the state of POPUP suppression is left unchanged.

If neither \( x: \) nor NOLOG is specified then the state of logging is left unchanged.

An application may temporarily disable exception popup messages by use of the DosError API. Use of yhis will not affect logging.

TRAPLOG was made available at fix pack 29 for OS/2 Warp V3.0 and base releases of later versions of OS/2.

SYSDUMP

The SYSDUMP command forces a System Dump to be initiated immediately regardless of the TRAPDUMP settings.

Warning:

The initiation of a System Dump causes an immediate termination of the system without any shutdown. No file system shutdown is
performed. The system behaves as if a fatal crash has occurred thus under rare circumstances data can be lost.

**Syntax.**

```
SYSDUMP [/NOPROMPT]
```

**Parameters**

`/NOPROMPT`  The user will not be prompted for a confirmation to proceed with initiating a system dump. The default behaviour is to prompt for confirmation with following message:

"Do you want to force a system dump? (Y/N)"

**Remarks**

The `SYSDUMP` command may be used from within a CMD file to automate the creation of a system dump.

--------------------------------------------

**OS/2 RAS Application Programming Interfaces**

This chapter describes the subset of OS/2 RAS APIs for use by application programmers, which are not described in the OS/2 Technical Library, Control Programming Reference.

**CAUTION:**

Some RAS Programming interfaces may be specific to a particular release of OS/2 or have release specific function.

The APIs discussed in this section are:

- `DosSysTrace`
- `DosGetSTDA`
- `DosForceSystemDump`
- `DosDumpProcess`
- `DosSuppressPopUps`
- `DosQueryRASInfo`
- **16-bit Error Logging APIs:**
  - `DosLogRegister`
  - `DosLogEntry`
  - `DosLogRead`
- **32-bit Error Logging APIs:**
  - `LogOpen`
  - `LogClose`
  - `LogAddEntries`

--------------------------------------------
DosSysTrace (Static Trace Event Recording)

Static trace recording is available as both an API and a DevHlp routine.

Select one of the following:

- DosSysTrace
- DevHlp_RAS

DosSysTrace (Add a Trace Record to the System Trace Buffer)

**DosSysTrace** allows a subsystem or system extension to add information to the system trace buffer.

**Note:** **DosSysTrace** is a 16-bit API.

**Coding Examples.**

```asm
EXTRN DosSysTrace:FAR

PUSH    WORD    MajorCode   ; major trace event code (240-255)
PUSH    WORD    Length      ; length of the variable length
                       ; area to be recorded (0-512)
PUSH    WORD    MinorCode   ; minor trace event code (0-255)
PUSH@    OTHER   Data       ; pointer to the area to be traced
                       ; (address parameter)
CALL    DOSSYSTRACE

16-bit MASM Example

APIRET16 APIENTRY16 DosSysTrace(USSHORT MajorCode, USHORT Length,
                                  USHORT MinorCode, PCHAR pData);

32-bit code Example using CSet/2
```

**Parameters.**

- **MajorCode**
  
  The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

- **Length**
  
  The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.

- **MinorCode**
  
  The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

- **pData**
  
  The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

**Results.**
**DosSysTrace** returns the following values:

0  
**NO_ERROR**

150  
**ERROR_SYSTEM_TRACE** (Trace is disabled for that event)

IF AX = 0  
Data traced
ELSE  
AX = Error_System_Trace  
Data not traced

**Note:** An example of when data would not be traced is if the major event code is not currently selected for tracing.

**Remarks.**

All trace records consist of a header and optional data. The header record is built by **DosSysTrace** and contains:

- Major event code
- Minor event code
- Process ID of caller
- Time stamp when the time is different from the previous trace record
- Flag field
- Data field (optional)

The optional data field contains the variable-length data as passed by the caller.

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The trace facility checks this array each time it is called to ensure that the major event specified is currently enabled for tracing. The array is located in the **Global Information Segment**.

A prototype definition for **DosSysTrace** may be found under **RAS API Prototypes**.

--------------------------------------------

**DevHlp_RAS (Add a Trace Record to the System Trace Buffer)**

The **DevHlp_RAS** function provides a service for device drivers to add information to the System Trace buffer.

**Note:** DevHlp_RAS is a 16-bit API.

**Coding Example.**

```
MOV AX, MajorCode ; major trace event code (240-255)
MOV BX, Length   ; length of data area (0-512 bytes)
MOV CX, MinorCode ; minor trace event code (0-255)
LDS SI, pData    ; pointer to trace data
MOV DL, 28H      ; DevHlp_RAS function code
CALL [Device_Help] ; invoke device helper
```

16-bit MASM Example

**Parameters.**

- **MajorCode**  
The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

- **Length**  
The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes...
will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.

MinorCode

The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

pData

The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

Results.

If CF = 0
   Trace record placed in trace buffer
Else
   Data not traced

The possible errors are:

- Tracing suspended
- Minor code not being traced
- PID not being traced
- Trace overrun

Remarks.

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The device driver must check this array before calling DevHlp_RAS to ensure that the major event specified is currently enabled for tracing. This array is located in the Global Information Segment.

All registers are preserved. Interrupts are disabled while the trace data is saved and then re-enabled if they were initially enabled.

DosGetSTDA (Get The System Trace Data Area)

The DosGetSTDA API is a 16 bit API that returns a copy of the system trace buffer (STDA).

Syntax

The following 16 bit C language function prototype can be used to call the DosGetSTDA API:

```
// 16 bit compiler
extern unsigned far pascal DosGetSTDA( SEL, SHORT, SHORT );

// 32 bit compiler
APIRET16 APIENTRY16 DosGetSTDA( SEL, SHORT, SHORT );
```

Where:

- SEL is the selector to the private buffer
- SHORT is the offset to the private buffer
- SHORT is the size of the buffer (maximum value = 64KB) records

Returns:

- 0 - indicates correct operation, buffer is now filled with copy of the system trace buffer
- ERROR_SYSTEM_TRACE - System trace is not enabled

Linker Considerations

In order to successfully resolve DosGetSTDA function calls in your program, the following lines must be added to the Linker Definition (DEF) file:
**Remarks**

DosGetSTDA returns a buffer that contains a copy of the system trace buffer. The buffer is circular with a header record that contains pointers to the first and last data bytes and a pointer to the next byte that was available for writing (the buffer is a snapshot of the system trace buffer at the time that the API was called). A set of trace records follows the header. Each trace record contains a trace event trailer and optionally a timestamp and/or a data field. A Timestamp record is optional and will only exist if bit 2 of the flag field in the Trace Event Trailer is set to OFF.

The trace event data contains the information describing each individual trace event. The events traced may be from OS/2 system supplied or other user supplied tracepoints. In either case the data is dependent on each individual tracepoint. Descriptions of the data and formatting instructions for the OS/2 system supplied tracepoints can be found in the System Tracepoints Reference chapter.

--------------------------------------------

**Trace Buffer Structures**

**Note:**

The from OS/2 2.11 fix pack 91 and OS/2 3.0 fix pack 8 the format of the STDA has changed to allow more meaningful time-stamp information. See New STDA Format at the end of this section for details.

**Circular Trace Buffer (STDA)**

<table>
<thead>
<tr>
<th>Pointer to</th>
<th>Pointer to</th>
<th>Pointer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>First (#1)</td>
<td>Next (#3)</td>
<td>Last (#2)</td>
</tr>
<tr>
<td>Entry #3</td>
<td>Entry #10</td>
<td></td>
</tr>
<tr>
<td>SYSTRACE</td>
<td>.</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pointer 1</th>
<th>Pointer 2</th>
<th>Pointer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Flags</td>
<td>Process ID</td>
</tr>
<tr>
<td>Major Code</td>
<td>Minor Code</td>
<td>Length</td>
</tr>
<tr>
<td>Length</td>
<td>Data (2 bytes)</td>
<td></td>
</tr>
</tbody>
</table>

**Field Descriptions:** Trace Control Record

<table>
<thead>
<tr>
<th>Name</th>
<th>#Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of Data Area</td>
<td>8</td>
<td>Contains ASCII 'SYSTRACE'</td>
</tr>
<tr>
<td>Pointer 1</td>
<td>2</td>
<td>Offset of first byte of the trace buffer</td>
</tr>
<tr>
<td>Pointer 2</td>
<td>2</td>
<td>Offset of last byte of the trace buffer</td>
</tr>
<tr>
<td>Pointer 3</td>
<td>2</td>
<td>Offset of next available byte in the trace buffer</td>
</tr>
</tbody>
</table>

**Field Descriptions:** Trace Event Trailer Record (with Timestamp)

<table>
<thead>
<tr>
<th>Name</th>
<th>#Bytes</th>
<th>Description</th>
</tr>
</thead>
</table>
Timestamp 2 Timestamp in seconds and hundredths of seconds (Conditional on bit 1 in the Flags byte)

Flags 1 Trace record flag
   Bit 0: 0 indicates an internal kernel generated trace record.
   Bit 1: 0 indicates that a time-stamp is present.
   Bit 2: 1 signifies that the trace record was generated in protect mode.
   Bit 3: 0 signifies a static trace record, 1 a dynamic trace record.
   Bit 4: 1 indicates an incomplete dynamic trace record.
   Bit 5 - 7: reserved.

PID 2 ID of the process calling the API being traced

Minor Code 2 Minor Event Code

Length 2 Length of data for the traced API

Major Code 1 Major Event Code

**Remarks**

The buffer returned by **DosGetSTDA** is a simple circular buffer that is a snapshot of the OS/2 System Trace buffer at the time that the API was called. The actual System Trace buffer is emptied by the call. The buffer contains a header record that has pointers to the FIRST, LAST and NEXT bytes in the buffer. The offsets of the FIRST and LAST bytes are constant and the offset to NEXT is used to indicate the last (most recent) trace record in the buffer. This pointer is logically moved backwards as the buffer is traversed. Since it is possible for a trace record to wrap back to the end of the buffer, it is necessary to look at each part of the data individually (trailer, timestamp and data) to determine whether the length of the data is greater than the distance between NEXT and FIRST. If the length is greater, then the data is continued at the offset to LAST.

For example (see figure below), the buffer has been traversed until the pointer to NEXT is at byte 26. The event trailer record is 8 bytes and the distance from NEXT to FIRST is 12, so the trailer is in contiguous memory. The pointer to NEXT is then set to byte 18. There is a timestamp which is two bytes. Our distance to FIRST is now 4 so the timestamp is contiguous and the pointer to NEXT is reset to 16. This record has 4 bytes of data attached to it. The distance to FIRST at this point is 2, so the data is wrapped: 2 bytes are adjacent to the NEXT, and the other 2 bytes begin at the pointer to LAST.

<table>
<thead>
<tr>
<th>First (Byte 14)</th>
<th>Next (byte 26)</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE HEADER</td>
<td>T EVENT TRAILER</td>
<td>T EVENT TRAILER (other trace</td>
</tr>
<tr>
<td>(14 BYTES)</td>
<td>DATA S DATA LENGTH: 4</td>
<td>S DATA LENGTH: 0 records)</td>
</tr>
<tr>
<td>2 Bytes</td>
<td>2 Bytes</td>
<td></td>
</tr>
</tbody>
</table>

End of data in the trace buffer is indicated by a trace event trailer that contains a major code field of zero and a length field of zero.

The display format of the OS/2 system supplied tracepoint data is described in the **System Tracepoints Reference** chapter. Note that for data using the ‘%S’ (ASCIIZ string) format type, the first byte of the data is reserved, bytes two and three contain the actual length of the string and the string begins at byte 4.

**TRACEFMT** Unformatted Trace Buffer

The trace formatter (TRACEFMT) is able to save the unformatted STDA buffer for formatting at a later date. The format of this buffer is as follows:

<table>
<thead>
<tr>
<th>File Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDA LN</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
STDA

SYSTRACE STA END NXT TRACE RECORDS......

1a              22 24 26 28

Remarks

STDALN

ULONG length of the STDA read by DosGetSTDALN. Length is 1 greater than then STDA end offset.

DATETIME

A DATETIME structure returned by DosGetDateTime when this file buffer is created.

CHECK KEY

The DATETIME field Exclusively ORed with the string constant "TRCFMTBUFF$".

STDA

The STDA returned by DosGetSTDALN.

Note:

DosGetSTDALN resets the internal start, end and next offsets after the STDA has been read. This allows trace formatting programs to detect an empty buffer.

For GA OS/2 2.x and 3.x the default start offset is 0x000e.

After fix pack 91 (OS/2 2.11) and fix pack 8 (OS/2 3.0) the the default start offset is 0x001e.

--------------------------------------------

New STDA Format

From fix pack 8 (OS/2 3.0) and fix pack 91 (OS/2 2.11) the system trace was enhanced to provide improved time-stamp information. Each trace records is time-stamped in hours, minutes, seconds and 1/100 seconds. The trace logging start and stop times are also logged and displayed by the TRACERMT command.

The spare bytes between the end of the STDA header and first trace record have been reserved for storing trace start and stop times. These are of the following format:

STDA

y y m d h n s c Y Y M D H N S C TRACE RECORDS......

28              30              38

Where:

yymdhnscc is the TRACE ON date and time in years, months, days, hours, seconds and 100th seconds.

YYMDHNSCC is the TRACE OFF date and time in years, months, days, hours, seconds and 100th seconds.

The time-stamp field of a trace record in now 4 bytes and contains in addition hours and minutes. The following diagram compares the different trace records of both old and new formats:

Old format with time-stamp:

Optional Data s h f Pid Min Len Maj
Old format without time-stamp:

Optional Data  f Pid Min Len Maj
-8  -4  0 1  3  5  7

New format with time-stamp:

Optional Data  H M s h f Pid Min Len Maj
-8  -4  0 1 2 3 4 5 7 9  a

where:
Optional Data is trace data of length specified by the Len field.
H is time in hours.
M is time in minutes.
s is time in seconds.
h is time in 1/100 seconds.
Pid is the process id under which the entry was logged. Zero implies interrupt time.
Min is the minor code.
Len is the length of the optionally traced data.
Maj is the major code.

Notes:
The format of the buffer used by TRACEFMT has not changed. Thus, as long as a correct header is appended to the extracted STDA then the new TRACEFMT will format the traced data.
The new TRACEFMT will also format trace data of the older format.
STDAs of the new and old formats may be distinguished by the value of the start offset in the STDA header:
For the old format this is 0x000e.
For the new format this is 0x001e.

DosForceSystemDump (Force a System Stand Alone Dump)

DosForceSystemDump allows an application to initiate a stand-alone system dump.

Syntax

APIRET APIENTRY DosForceSystemDump(ULONG reserved);

32-bit code Example using CSet/2
Parameters
reserved  Reserved doubleword field that is set to 0L.

Returns.
There is no return from this API.

Remarks.
There is no return from this API. The system is halted abruptly and a stand-alone dump is initiated. After the stand-alone dump process has completed the system must be re-booted.

No shut down activity is performed when this API is called. File system buffers are not written to disk, cache is not flushed and files are not closed. Data loss may result.

_DosForceSystemDump_ is equivalent to using the _Ctrl-Alt-Numlock-Numlock_ key sequence.

C Language prototype definitions for the _DosForceSystemDump_ API may be found under _RAS API Prototypes_.

To format a system dump see _The Dump Formatter User Guide_.

For related information see:
- TRAPDUMP CONFIG.SYS command
- CREATEDD command in the OS/2 Command Reference

DosDumpProcess (Enable/Disable ProcessDump)

_DosDumpProcess_ allows an application:
- to enable or disable dynamically the Process Dump Facility.
- to force a process dump for a given process.

The default setting is for Process Dump to be disabled unless overridden by the _DUMPPROCESS CONFIG.SYS command_.

Syntax
APIRET APIENTRY DosDumpProcess(ULONG Flag, ULONG Drive, PID pid);

32-bit code Example using CSet/2

Parameters
Flag
Doubleword field that may take one of the following values:
- (DDP_DISABLEPROCdump 0x00000000L) disable process dumps
- (DDP_ENABLEPROCdump 0x00000001L) enable process dumps to be taken to a file in the root directory of a drive specified by the _Drive_ parameter.
- (DDP_PERFORMPROCdump 0x00000002L)

Drive
Doubleword containing the ASCII value of the drive letter to which the PDUMP.nnn dump files will be written when DDP_ENABLEPROCdump is specified. For DDP_DISABLEPROCdump this parameter is ignored.

pid
Doubleword containing the process Id of the process to be dumped.
This option is valid only with DDP_PERFORMPROCDUMP. If zero is specified for Pid then the current process is dumped.

**Returns.**

Return Code.

**DosDumpProcess** returns the following values:

- **0**  
  NO_ERROR
- **87**  
  ERROR_INVALID_PARAMETER
- **303**  
  ERROR_INVALID_PROCID

**Remarks.**

When Process dump is enabled a dump file is written whenever a ring 3 process traps. The file takes a name **PDUMP.nnn** where **nnn** is incremented sequentially (starting from 000) for each successive dump.

The directory to which PDUMP.nnn will be written is always the root directory of **Drive**.

C Language prototype definitions for the **DosDumpProcess** may be found under RAS API Prototypes.

The content of a Process Dump comprises register information at time of trap, system control blocks (TCB, TSD, PTDA, MTE, SMTE, OTE, VMAR, VMOB, LTD) that describe the state of the process at the time of error, ring 0 and ring 3 stack data for the trapping process.


**Note:**

DDP_PERFORMPROCDUMP is not available in some early releases of OS/2 V2.11

--------------------------------------------

**DosSuppressPopUps (Suppress Trap Exception Pop-Up Messages)**

**DosSuppressPopUps** allows an application to enable or disable dynamically Trap Exception pop-up suppression and to specify the drive where the pop-up suppression log will be recorded.

The default setting is for disabled pop-up suppression unless overridden by the **SUPPRESSPOPUPS CONFIG.SYS command**.

**Syntax**

```c
APIRET APIENTRY DosSuppressPopUps(ULONG Flag, ULONG Drive);
```

32-bit code Example using CSet/2

**Parameters**

- **Flag**  
  Doubleword field that may take one of the following values:
  - (SPU_DISABLESUPPRESSION 0x00000000L)  
    Disable pop-up suppression
  - (SPU_ENABLESUPPRESSION 0x00000001L)  
    Enable pop-up suppression and pop-up logging to file POPUPLOG.OS2 on drive specified by the **Drive** parameter.
Drive

Doubleword containing the ASCII value of the drive letter to which the POPUPLOG.OS2 log file will be written when SPU_ENABLESUPPRESSION is specified. With SPU_DISABLESUPPRESSION, Drive is ignored.

Returns.

Return Code.

DosSuppressPopups returns the following values:

0
NO_ERROR

87
ERROR_INVALID_PARAMETER

Remarks.

The directory to which POPUPLOG.OS2 will be written is always the root directory of Drive.

A prototype definition of DosSuppressPopUps may be found under RAS API Prototypes.

See also DosError API in the Control Program Programming Reference.

--------------------------------------------

DosQueryRASInfo (Query RAS Information)

DosQueryRASInfo returns information about active trace event recording and System Logging facility from the Global Information Segment (InfoSegGDT). dump.

Syntax

APIRET APIENTRY DosQueryRASInfo(ULONG Index, PPVOID Addr);

32-bit code Example using CSet/2

Parameters

Index

Doubleword field that may take one of the following values:

• (SPU_SIS_MEC_TABLE 0x00000001L)

Return the address of the table of actively traced major event codes in the InfoSegGDT. The table is 32 bytes long, each bit represents each major event code from 0 to 255.

• (SIS_SYS_LOG 0x00000002L)

Return the address of the SYSLOG status word from the InfoSegGDT. The status may contain a combination of:

- (LF_LOGENABLE 0x0001) Logging enabled
- (LF_LOGAVAILABLE 0x0002) Logging available

Returns.

Return Code.

DosQueryRASInfo returns the following values:

0
NO_ERROR

5
ERROR_ACCESS_DENIED

87
Remarks.

For related information see:

• Logging Facility
• The OS/2 Trace Facility

--------------------------------------------

16 Bit Error Logging API's for IBM OS/2 Version 2.1

This section describes the "Logging Facility for OS/2 2.1". This comprises a set of 3 APIs, the logging daemon (LOG.SYS) and the log formatter (SYSLOG).

Both the Logging Daemon and Log Formatter are described in the OS/2 Command Reference - see LOG.SYS under DEVICE statement of CONFIG.SYS and the SYSLOG command.

Note:

C Language prototype definitions for the Error Logging APIs may be found under RAS API Prototypes.

The following topics are described in this section:

• Static vs Dynamic Error Log Record I.D. Registration
• DosLogRegister API
• DosLogEntry API
• DosLogRead API
• Error Log Entry Formatting DLL Routines

--------------------------------------------

Dynamic vs. Static Error Log Record I.D. Registration

OS/2 2.0 users of the DosLogEntry API will not need to use the DosLogRegister API. The DosLogRegister API is only maintained on OS/2 2.0 to support existing OS/2 1.3 programs that did need to use the API.

The OS/2 2.0 version of the DosLogRegister API will always return a "default" Error Log record I.D.. It will accept a format template string as an input, but it will do nothing with the string since format template strings will not be saved within the OS/2 2.0 version of the Error Log file.

The OS/2 2.0 version of the DosLogEntry API will behave similarly to the OS/2 1.3 version of the API. Since the OS/2 2.0 version of the system Error Logging facility no longer supports the saving of format template strings within the Error Log file, it is necessary to provide a method by which DosLogEntry callers can associate their Error Log entry with a formatting (.DLL) routine. The OS/2 2.0 version of the DosLogEntry API will make a special interpretation of the Originator Name field within the packet header. It will be assumed that this name field (if not NULL) contains the name of a Error Log formatting .DLL module.

--------------------------------------------

DosLogRegister

There are two major differences between the OS/2 2.0 version of DosLogRegister and the 1.3 version of the API:

• DosLogRegister no longer supports dynamic registration of Error Log record I.D.’s. Instead, the API always returns a single
DosLogRegister no longer supports entry format template registration. While the API still accepts a format template as part of its input data packet, the format template will not be acted upon in any way.

DosLogRegister continues to support the existing alert notification registration function.

The description of the OS/2 2.0 version of the DosLogRegister API follows:

**Syntax**

APIRET16 APIENTRY16 DosLogRegister((PUSHORT) LogHandle, (PVOID) LogRegList, (PUSHORT) RequestID)

32-bit code Example using CSet/2

**Parameters**

**LogHandle**

The address of the word in which the system will return the handle of a named pipe that will be transparently used in subsequent DosLogRead calls.

**LogRegList**

The address of the log registry buffer.

**RequestID**

The address of the word that the system will fill in with a "default" Error Log record I.D. (if the 'Error Log record I.D.' field in the log registry buffer is set by the caller to -1)

**Returns**

Return code

DosLogRegister returns the following values

0 Success
non-zero Failure.

Possible reasons for failure are:

Facility unavailable
Record I.D. in use
Registration failed (general failure)
Invalid I.D.
Too many open files
Too many semaphores
Semaphore not found
User semaphore limit reached
Request timed out without satisfaction
Error Log buffer temporarily full

**Remarks**

Log Registry Buffer format description:

length of the registration data 2
reserved                                     2
Error Log record I.D.                        2
offset to the format template layout field   2
semaphore name string          variable length
format template layout         variable length

Where:

'length of the registration data'
    is the total number of bytes in the current Log Registry Buffer (this length includes the two byte length field itself)

'reserved'
    is a two byte reserved field

'Error Log record I.D.'
    contains the Error Log record I.D. that caller wishes to be registered for. If the field is set to 0xFFFF (-1), then a
    "default" record I.D. is returned in the word pointed to by the 'RequestID' parameter. This field can be used to specify
    an alert notification record I.D. (that is, the caller wishes to be alerted whenever an Error Log Entry containing this
    record I.D. is logged).

'offset to the format template layout field'
    is the offset within the Log Registry Buffer to the start of the format template layout area.

'semaphore name string'
    is the name of a system semaphore, created with the nonexclusive option, that will be used to alert the caller's
    process when an Error Log entry containing the specified 'Error Log record I.D.' is logged. The name string is an
    ASCIIZ string.

'format template layout'
    is an area within the Log Registry Buffer that contains the formatting structure information that is placed within the 1.3
    Error Log file. This area is not used in the OS/2 2.0 version of the DosLogRegister call. However, the 'length of the
    registration data' field should reflect the size of this area.

In order to resolve successfully DosLogRegister function calls in your program, the following lines must be added to the Linker Definition
(DEF) file:

IMPORTS
DOSLOGREGISTER=DOSCALL1.195

DosLogEntry

There are two major differences between the OS/2 2.0 version of DosLogEntry and the 1.3 version of the API:

- Since the DosLogRegister API will only return a "default" Error Log record I.D. to its caller, the DosLogEntry caller must
  override this "default" record with the appropriately statically allocated record I.D. if the caller wishes to see the "correct" record
  I.D. in the Error Log record.

- Since there is no explicit "Error Log record formatting DLL module name" field in the DosLogEntry log data packet, the API will
  attempt to interpret the 'Originator Name' field in the packet's header portion as a formatting DLL module name.

The description of the OS/2 2.0 version of the DosLogEntry API follows:

Syntax

APIRET16 APIENTRY16 DosLogEntry((USHORT) Function,
Parameters

Function specifies the type of log entry:

0H    Reserved
1H    Error Logging
2H-FFFFH    Reserved

LogData is the address of the log data buffer that contains one or more variable length log packets.

Returns

Return Code.

DosLogEntry returns the following values:

0    Success
non-zero    Failure

Possible reasons for failure:

Invalid function
Facility unavailable
Facility suspended
Error Log buffer temporarily full

Remarks

Error Log Data Buffer format description:

Multiple log packets can be included within a single log data buffer. In the following diagram, the size of each field is indicated in bytes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td># of log packets (within the buffer)</td>
<td>2</td>
</tr>
<tr>
<td>length of the current log packet</td>
<td>&lt;2</td>
</tr>
<tr>
<td>Error Log record I.D.</td>
<td>2</td>
</tr>
<tr>
<td>time of logging</td>
<td>4</td>
</tr>
<tr>
<td>date of logging</td>
<td>4</td>
</tr>
<tr>
<td>originator name</td>
<td>8</td>
</tr>
<tr>
<td>qualifier name</td>
<td>4</td>
</tr>
<tr>
<td>Error Log entry data</td>
<td>&lt;=1024 &lt;</td>
</tr>
</tbody>
</table>

Where:

'\texttt{# of log packets}' is the number of separate packets contained within the user's buffer

'\texttt{length of the current log packet}'
is the number of bytes in the current log packet within the user's log data buffer (this length includes the length of all the log packet control fields and the size of the Error Log entry data).

'Error Log record I.D.'
is the record I.D. for the current Error Log entry (I.D. registration will be statically registered by the OS/2 development organization). The caller may pass in the "default" Error Log record I.D. that is returned by the DosLogRegister API.

'time of logging'
is filled in by the system Error Logging facility

'date of logging'
is filled in by the system Error Logging facility

'originator name'
is a primary name field that is provided by the caller

'qualifier name'
is a secondary name field that is provided by the caller

'Error Log entry data' is an optional variable length set of data that can be supplied by the caller (the format of the data is under the control of the caller).

In order to successfully resolve DosLogEntry function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```plaintext
IMPORTS
DOSLOGENTRY=DOSCALL1.193
```

--------------------------------------------

DosLogRead

The description of the OS/2 2.0 version of the DosLogRead API follows:

```plaintext
APIRET16 APIENTRY16 DosLogRead((USHORT) LogHandle,
    (USHORT) Length,
    (PVOID) LogBuffer,
    (PUSHORT) ReadSize)
```

32-bit code Example using CSet/2

**Parameters**

LogHandle
is the named pipe handle returned by DosLogRegister()

Length
is the length (in words) of the caller's log buffer

LogBuffer
is the address of the caller's buffer, into which the system Error Logging facility will place a single Error Log entry packet (formatted in the manner of the 16 bit DosLogEntry API).

ReadSize
is the address of a word, into which the system Error Logging facility will place the number of bytes that it wrote into the caller's log buffer. If a zero is returned here, then there was no Error Log packet to return.

**Returns**

Return code
**DosLogRead** returns the following values:

- **0** indicating Success.
- **non-zero** indicating error

Possible reasons for failure:

- Invalid log handle
- Facility unavailable
- Buffer too small

In order to resolve successfully **DosLogRead** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```plaintext
IMPORTS
DOSLOGREAD=DOSCALL1.196
```

**DosLogRead** returns Error Log entries that are formatted in the manner of the 16 bit **DosLogEntry** API.

---

**Error Log Entry Formatting DLL Routines**

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the **DosLoadModule** API to create a run-time link to the specified formatting DLL module. It uses the **DosFreeModule** API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of "colliding" with another DLL module of the same name, it is suggested that a formatting DLL module be labelled with a name that adheres to the following standard form:

```plaintext
ELGxxxxx.DLL
```

(where "xxxxx" corresponds to the Error Log record I.D. (in decimal) of any one of the types of records that the formatting routine is designed to handle)

for example, "ELG00127.DLL" is a standardized name for a formatting DLL module that recognizes (among other things) Log records with I.D. of 127 (decimal)

This standard naming convention is suggested because it is assumed that the Error Log records of any one I.D. will only be recognized by a single formatting routine. Therefore the use of the "xxxxx" suffix (based on record I.D.) should assure uniqueness for the formatting module name.

The static Error Log record I.D. registration mechanism that is enforced by the OS/2 development organization will attempt to keep a list not only of the Error Log record I.D.’s in use, but also the names of the formatting DLL modules that correspond to each record I.D.. This will also help to reduce the possibility of formatting DLL module names "colliding".

In addition to its single formatting routine, each formatting DLL module must contain a global variable named "ELOG_FORMAT". For OS/2 2.0, this exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it attempts to
access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable will in future releases be used as a revision level for the SYSLOG/formatting DLL module interface specification.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel. The SYSLOG utility will be written to use the "ordinal" version of the DosGetProcAddr API.

Error Log record formatting DLL routines must be written as 32 bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```
ULONG ELGxxxxx((PVOID) Log_Record, (PVOID) String_Buffer,  
                (ULONG) Buffer_Length, (PULONG) String_Length)
```

**Parameters**

- **Log_Record**: a linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled "Error Log File Entry Format", except that the linear pointer points to the "TOT_LENGTH" field (since the "PREV_PTR" and "PREV_SIZE" fields are of no interest to a formatting routine).

- **String_Buffer**: is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string "line" within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

- **Buffer_Length**: is a 32 bit integer that contains the maximum size of the the 'String_Buffer'.

- **String_Length**: is a pointer to a 32 bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in 'String_Buffer'.

**Returns**

- **ELGxxxxx** returns the following:
  - *0* indicating success.
  - *-1* indicates insufficient space in 'String_Buffer' positive values indicate formatting routine errors.

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the 'String_Buffer', then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller's output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger 'String_Buffer'.

SYSLOG will contain logic to format the standard SNA Generic Alert entry (that is, Error Log record I.D. of 2). This is necessary since most of the existing Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record I.D. of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

--------------------------------------------

32-Bit Error Logging API's for IBM OS/2 Version 2.1 and 3.0
This section describes the "Logging Facility for OS/2 2.1 and 3.0". This comprises a set of 4 APIs, a DevHlp function, the logging deamion (LOGDAEM.EXE), logging device driver (LOG.SYS) and the log formatter (SYSLOG).

The Logging Deamion, Device Driver and Log Formatter are described in the OS/2 Command Reference - see LOG.SYS under DEVICE statement of CONFIG.SYS and the SYSLOG command.

Note:

C Language prototype definitions for the Error Logging APIs may be found under RAS API Prototypes.

The following topics are described in this section:

- LogOpen API
- LogClose API
- LogAddEntries API
- 32-bit Error Log Entry Formatting DLL Routines
- DevHlp_LogEntry Device Driver interface

The set of four 32-bit logging APIs provide equivalent functionality to the three 16-bit logging APIs discussed in the previous section. They may be used as a complete replacement to the 16-bit set.

LogOpen

LogOpen is a 32-bit system Error Logging facility high level API. It is used to open a connection to the facility (through the System Logging Service device driver).

The description of the LogOpen API call follows:

Syntax

APIRET APIENTRY LogOpen(PHFILE phf);

Parameters

phf
points to a file handle holder that on return will hold an open file handle

Returns

Return code.

LogOpen returns the following values:

0
Success.

non-zero
Facility not available.

Remarks

The file handle that is returned by the LogOpen API is required in all subsequent high level system Error Logging facility API calls.

In order to resolve successfully LogOpen function calls in your program, the following lines must be added to the Linker Definition (DEF) file:
**LogClose**

**LogClose** is a 32-bit system Error Logging facility high level API. It is used to close a connection to the facility.

The description of the **LogClose** API call follows:

**Syntax**

```c
APIRET APIENTRY LogClose(HFILE hf);
```

**Parameters**

- **hf** is the file handle returned by **LogOpen()**

**Returns**

Return code.

- **LogClose** returns the following values:
  - 0 Success.
  - non-zero Failure. Possible reason: facility not open.

**Remarks**

In order to resolve successfully **LogClose** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```c
IMPORTS
LogClose=DOSCALL1.431
```

**LogAddEntries**

**LogAddEntries** is a 32-bit system Error Logging facility high level API. It is used to allow application processes to add Error Log entries to the internal Error Log buffer that is maintained by the System Logging Service device driver.

The description of the **LogAddEntries** API call follows:

**Syntax**

```c
```

**Parameters**

**Remarks**

In order to resolve successfully **LogAddEntries** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```c
IMPORTS
LogAddEntries=DOSCALL1.432
```

**Remarks**

In order to resolve successfully **LogAddEntries** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:
APIRET APIENTRY LogAddEntries(HFILE hf, ULONG service, 
PVOID log_data_address);

**Parameters**

hf is the file handle returned by LogOpen()

service specifies the class of logging facility:

- 0x0 Reserved
- 0x1 Error Logging
- 0x2 - 0xffff Reserved

log_data_address is the address of a buffer that contains a variable length Error Log entry. The first word of the buffer contains the number of packets in the Error Log entry

**Returns**

Return code.

LogAddEntries return the following values:

- 0 Success
- non-zero Failure

Possible reasons for failure are:

- Invalid log type
- Facility unavailable
- Facility suspended
- Facility not open
- Error Log buffer temporarily full

**Remarks**

Error Log Entry Buffer format description:

Multiple Error Log packets can be included within a single Error Log entry buffer. If multiple packets are included within a single buffer, each individual packet should be aligned on a double word boundary. In the following diagram, the size of each field is indicated in bytes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>packet revision number</td>
<td>2</td>
</tr>
<tr>
<td># of Error Log entry packets</td>
<td>2</td>
</tr>
<tr>
<td>length of this Error Log entry packet</td>
<td>2</td>
</tr>
<tr>
<td>Error Log record I.D.</td>
<td>2</td>
</tr>
<tr>
<td>status flags</td>
<td>4</td>
</tr>
<tr>
<td>qualifier name</td>
<td>4</td>
</tr>
<tr>
<td>reserved</td>
<td>4</td>
</tr>
<tr>
<td>time of logging</td>
<td>4</td>
</tr>
</tbody>
</table>

Within a
Where

'packet revision number' is an integer value that can be used to distinguish error logging packets that are intended to be handled by different revisions of the LogAddEntries API. For the initial version of the API, this field should be set to a value of 1. This field is included in the packet to support future backward compatibility.

'# of Error Log entry packets' is the number of separate packets contained within the user's buffer.

'length of this Error Log entry packet' is the number of bytes in the current Error Log entry packet within the user's Error Log Entry Buffer (this length includes the length of all the Error Log entry packet control fields and the size of the Error Log entry text). To support efficient logging execution, this length should be a multiple of 4 bytes (i.e. if necessary the user should pad the Error Log entry packet).

'Error Log record I.D.' is the record I.D. for the current Error Log entry (I.D. registration will be statically registered by the OS/2 development organization).

'status flags' is a two byte flag holder that contains three single bit flags:

(BIT 0) is used to indicate whether the current Error Log entry packet contains space in which the Error Logging facility can place a long process name ("on" indicates YES, "off" indicates NO);

(BIT 1) is used to indicate whether the current Error Log entry packet contains an 8 byte originator name or a 256 byte originator name ("on" indicates a 256 byte originator name, "off" indicates an 8 byte originator name);

(BIT 2) is used to indicate that the caller has placed time and date values in the Error Log entry packet and does not wish to have those values modified during the logging process ("on" indicates that the Error Log entry packet already contains time and date values, "off" indicates the packet does not already contain time and date values);

All the other 29 bits in 'status flags' are considered reserved at this time and will be zeroed by the LogAddEntries API.

'qualifier name' is a secondary name field that is provided by the caller

'reserved' is a four byte reserved field

'time of logging' is filled in by the system Error Logging facility (unless BIT 2 of the 'status flags' field is "on", indicating that the caller has preset a time value).

'date of logging' is filled in by the system Error Logging facility (unless BIT 2 of the 'status flags' field is "on", indicating that the caller has preset a date value);

'originator name' is a primary name field that is provided by the caller.

'process name' is an optional long process name field that will be filled in by the Error Logging facility if the field is provided by the caller in the Error Log entry packet.

'formatting DLL module name' is the optional name of a DLL module that houses a formatting routine that recognizes this type of Error Log entry data.
and can format it for display by the SYSLOG utility. The name is specified as an ASCIIZ string that can be up to eight characters in length. If no module name is specified in this field, then SYSLOG will display the data portion of the Error Log entry as a hexadecimal dump.

'Error Log entry data' is an optional variable length set of data that can be supplied by the caller (the format of the string is under the control of the caller).

The format and function of the LogAddEntries API call is very similar to that of the 16-bit DosLogEntry call. There are several functional differences from the DosLogEntry call:

- The user-supplied error log entry Record I.D. will now be a statically allocated value rather than a dynamically allocated value.
- The maximum size of the originator name field in the caller's packet has been increased from 8 bytes to 256 bytes. The caller can specify whether the packet contains an 8 byte originator name field or a 256 byte originator name field.
- The maximum size of the variable length data portion within the caller's packet has been increased from 1024 bytes to 3400 bytes
- The order of the fields within the Error Log entry has been slightly rearranged to support the creation of smaller internal control messages.

In order to resolve successfully LogAddEntries function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

IMPORTS
LogAddEntries=DOSCALL1.432

--------------------------------------------

32-Bit Error Log Entry Formatting DLL Routines

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the DosLoadModule API to create a run-time link to the specified formatting DLL module. It also uses the DosFreeModule API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of "colliding" with another DLL module of the same name, it is suggested that a formatting DLL module be labeled with a name that adheres to the following standard form:

ELGxxxxx.DLL (where "xxxxx" corresponds to the Error Log record I.D. (in decimal) of any one of the types of records that the formatting routine is designed to handle)

e.g. "ELG000127.DLL" is a standardized name for a formatting DLL module that recognizes (among other things) Error Log records with I.D. of 127 (decimal)

This standard naming convention is suggested because it is assumed that the Error Log records of any one I.D. will only be recognized by a single formatting routine. Therefore the use of the "xxxxx" suffix (based on record I.D.) should assure uniqueness for the formatting module name.

The static Error Log record I.D. registration mechanism that is enforced by the OS/2 RAS development group will attempt to keep a list not
only of the Error Log record I.D.’s in use, but also the names of the formatting DLL modules that correspond to each record I.D.. This may also help to reduce the possibility of formatting DLL module names “colliding”.

In addition to its single formatting routine, each formatting DLL module must contain a global variable named “ELOG_FORMAT”. This exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it will attempt to access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable may in future releases be used a sort of revision level for the SYSLOG/formatting DLL module interface specification. That is why it will initially be forced to a value of 1.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel.

Error Log record formatting DLL routines must be written as 32-bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```c
APIRET APIENTRY ELGxxxxx(PVOID Log_Record,
    PVOID String_Buffer,
    ULONG Buffer_Length,
    PULONG String_Length);
```

**Parameters**

Log_Record

is a linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled "Error Log File Entry Format", except that the linear pointer points to the “TOT_LENGTH” field (since the “PREV_PTR” and “PREV_SIZE” fields are of no interest to a formatting routine).

String_Buffer

is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string "line" within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

Buffer_Length

is a 32 bit integer that contains the maximum size of the the ‘String_Buffer’.

String_Length

is a pointer to a 32 bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in ‘String_Buffer’.

**Returns**

ELGxxxxx returns the following:

0

indicating success.

-1

indicates insufficient space in ‘String_Buffer’ positive values indicate formatting routine errors.

**Remarks**

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the ‘String_Buffer’, then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller's output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger ‘String_Buffer’.
SYSLOG contains logic to format the standard SNA Generic Alert entry (i.e. Error Log record I.D. of 2). This is necessary since most of the existing Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record I.D. of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

-------------------------------

DevHlp_LogEntry Device Driver Interface

DevHlp_LogEntry provides a device driver interface to the logging facility.

The description of the LogEntry DevHlp function follows:

Calling sequence -  LES BX, log_data_address
MOV CX, service
MOV DL, DevHlp_LogEntry  /* LogEntry function code 0x3b */
CALL [Device_Help]

Parameters

- **log_data_address** is the address of a buffer that contains a variable length Error Log entry. (See the section on the LogAddEntries high level API for further details.)

- **service** the class of logging facility:
  - 0x0 Reserved
  - 0x1 "Old-Style" Error Logging call ("old" 16-bit (DosLogEntry-style) data packet provided).
  - 0x2 - 0x2f Reserved for future use.
  - 0x80 - 0x8f Reserved for internal use by the System Logging Service device driver.
  - 0x90 "New_Style" Error Logging call ("new" 32-bit (LogAddEntries-style) data packet provided).
  - 0x91 - 0xffff Reserved for future use.

Returns

Return code in AX:

- 0 Success
- non-zero Failure.

Possible errors:

- Invalid log type
- Facility unavailable
- Facility suspended

Remarks

When CX is set to 80H, DS:SI is set to point to the device driver header block of the System Logging Service device driver.

-------------------------------
RAS API Prototypes

The following is a sample C language header file that contains sample prototype definitions for the RAS APIs.

/* definitions for DosDumpProcess */
#define DDP_DISABLEPROCDUMP 0x00000000L /* disable process dumps */
#define DDP_ENABLEPROCDUMP 0x00000001L /* enable process dumps */
#define DDP_PERFORMPROCDUMP 0x00000002L /* perform process dump */

/* definitions for DosSuppressPopUps */
#define SPU_DISABLESUPPRESSION 0x00000000L /* disable popup suppression */
#define SPU_ENABLESUPPRESSION 0x00000001L /* enable popup suppression */

/* definitions for DosQueryRASInfo Index */
#define SIS_MMIOADDR            0
#define SIS_MEC_TABLE           1
#define SIS_SYS_LOG             2
#define LF_LOGENABLE    0x0001          /* Logging enabled */
#define LF_LOGAVAILABLE 0x0002          /* Logging available */

APIRET  APIENTRY        DosQueryRASInfo(ULONG Index, PPVOID Addr);
APIRET  APIENTRY        DosForceSystemDump(ULONG reserved);
APIRET  APIENTRY        DosDumpProcess(ULONG Flag, ULONG Drive, PID Pid);
APIRET  APIENTRY        DosSuppressPopUps(ULONG Flag, ULONG Drive);
APIRET16 APIENTRY16    DosSysTrace(USHORT Majorcode, USHORT Length, USHOR
ALong Minorcode, PCHAR pData);

APIRET16 APIENTRY16 DosGetSTDA(SEL, SHORT, SHORT );

/* 32-bit Logging Facility Function Prototypes */

/* Logging Defines */

/* LogRecord status bits */
#define LF_BIT_PROCNAME     0x0001L
#define LF_BIT_ORIGIN_256   0x0002L
#define LF_BIT_DATETIME     0x0004L
#define LF_BIT_SUSPEND      0x0008L
#define LF_BIT_RESUME       0x0010L
#define LF_BIT_REDIRECT     0x0020L
#define LF_BIT_GETSTATUS    0x0040L
#define LF_BIT_REGISTER     0x0080L
#define LF_BIT_REMOTE_FAIL  0x0100L

/* Log Entry Record Header for 2.X */
/* This is format used by 2.0 device */
/* drivers and callers of LogAddEntries */
typedef struct LogRecord
{  USHOR
T len ; /* this record length(includes len field)*
USHOR
T rec_id ; /* record id */
ULONG    status ; /* record status bits(see LF_BIT_) */
UCHAR    qualifier[4] ; /* qualifier tag */
ULONG    reserved ; /* hours minutes seconds hundreds */
ULONG    date ; /* day month (USHORT)year */
UCHAR    data[1] ; /* begin of variable data that includes: */
}

#define ERRLOG_SERVICE        1L
#define ERRLOG_VERSION        1

/* 32-bit Logging Facility Function Prototypes */
OS/2 System Control Block Reference

This chapter contains details of some of the more important system control blocks used in debugging.

Where major differences in format exist between **ALLSTRICT** and **RETAIL**, and between versions of OS/2 then each version of the control block is given. Otherwise only OS/2 Warp V3.0 **ALLSTRICT** kernel versions of the control blocks are given and may be assumed to be applicable to also **RETAIL** and earlier versions of OS/2.

**Warning:**

The information given in this is for debugging purposes only. The layout of the control blocks may change from one release of OS/2 to the next. They are not to be considered a programming interface.

The following System Components are included in this chapter and an overview is provided in the next section: **Overview of Kernel Components and Interfaces**.

**Miscellaneous System Control Blocks**

This section describes system structures that are common to all components. These include: SAS and RMP.

**Semaphore Management**

This section describes the control blocks used for RamSem, FSRamSem Ksem, SysSem, PM/GRE, 32-bit, and MuxWait Semaphores.

**Memory Management**

This section describes the following control blocks used by Memory Management:
VMAL, VMOB, VMAR, VMCO, VMAT, VMAH, VMKH, PAI, PGDATA, PF and VP.

**Scheduler/Dispatcher**
This section describes the following control blocks used by Thread and Process Management:
- PTDA, TCB, TSD, ljmp, GISEG, LISEG, PIB, TIB, EXENT and Exception Handler structures.

**System Loader**
This section describes the following control blocks used by the System Loader component:
- MTE, SMTE, OTE, STE

**File System**
This section describes the following control blocks used by the File System component:
- SFT, MFT, FSC, RLR, VPB, DBP, CDS, BUF, Named and Anonymous Pipes.

**I/O and Device Driver**
This section describes the structures that relate to low level I/O. These include: Request Packets, BIOS Parameters Blocks and Device Driver Headers, Virtual Device Driver Entry Points.

-----------------------------

**Overview of Kernel Components and Interfaces**

The OS2KRNL modules lies at the heart of OS/2 - it is essentially operating system.

The kernel comprises an number of internal components, each responsible for a different aspect of running the system. It also has a number of interfaces that provides services to applications, device drivers and file systems.

These aspects are now considered in a little more detail and are summarised in the diagram shown under: The OS/2 Kernel's Interfaces (steady state).

-----------------------------

**Kernel Components**

**Task management and the Scheduler.**
This is responsible for thread and process management. The functions performed include:
- Thread and Process creation and termination.
- Thread scheduling (priority and state management).
- Preparing threads for dispatching.
- Blockin and Running.
- Implementing the Thread and Process related APIs.

The Scheduler's principle control blocks are:
- **PTDA** Per Task Data Area
- **TCB** Thread Control Block
- **TSD** Thread Swappable Data
- **TSS** Task State Segment (H/W)

**System Loader**
This is responsible for load module management. The Loader's principle responsibilities include:
- Bringing modules into memory and performing fixups.
Managing modules resources.
Managing dynamic linking.
Tracking module references.
Deleting modules from memory.
Managing the discarding and swapping of module pages.
Implementing module related APIs.

The Loader's principle control blocks are:

- MTE: Module Table Entry
- SMTE: Swappable Module Table Entry
- OTE: Object Table Entry
- STE: Segment Table Entry

Memory Management

Memory Management is responsible for managing physical, virtual, and swapper memory. Its principle roles include:

- Allocation and assignment of physical pages of memory.
- Allocation and assignment of virtual storage.
- Managing the swapper.
- Memory locking.
- Implementing memory related APIs.

The principle control blocks of Memory Management are:

- VMAR: Virtual Memory Arena Record
- VMOB: Virtual Memory Object Record
- PF: Page Frame Structure
- VP: Virtual Page Structure
- PTE: Page Table Entry (H/W)

File System

The File System kernel component responsibilities include:

- Access to FAT formatted media.
- Interfacing with File System Drivers for accessing non-FAT media.
- Managing and tracking the status of all open files.
- Path Management
- File sharing and serialisation.
- Providing helper Kernel services for FSDs.
- Implementation of all File System APIs.

The principle control blocks of the File System include:

- MFT: Master File Table Entry
- SFT: System File Table Entry
- CDS: Current Directory Structure
Device and I/O Management

This component is responsible for interfacing with Physical Device Drivers. Its responsibilities include:

- Routing requests to PDDs from applications
- Managing interrupts
- Providing helper kernel services for PDDs.

The principle control blocks for device management include:

- IRQI: IRQ Information Array
- DIRQ: Device IRQ Information
- REQ: PDD request Packet.
- DEV: PDD device header.

Virtual Dos Machine

This component is responsible for providing the entire Dos Machine emulation. This has not been covered in this book, except for the Virtual Device Driver interface.

Kernel Interfaces

The Kernel provides the following external interfaces:

Application (R3/2) Interface

Application access kernel services via GDT call gates. These are called either directly from the application program or via the DOSCALL1.DLL module, where additional Ring 3 processing is required before calling the kernel. Some system interfaces are able to be implemented entirely within Ring 2/3. In these cases, DOSCALL1.DLL does not make any kernel calls.

The kernel interfaces are represented by a fictitious module called DOSCALLS.DLL.

File System Driver (FSD)

The FSDs run in ring 0 as separately loaded modules. They are provided a set of interfaces to the kernel via the FSD_Hlp (File System Helper) calls.

Physical Device Driver (PDD)

The PDDs run in ring 0 as separately loaded modules. They are provided a set a interfaces to the kernel via the Dev_Hlp (Device Helper) calls.

Virtual Device Driver (VDD)

The VDDs run in ring 0 as separately loaded modules. They are provided a set a interfaces to the kernel via the VDD_Hlp (Virtual Device Driver Helper) calls.

Compatibility BIOS

The compatibility BIOS resides within the OS2LDR module. It provides a hardware implementation independent layer through which the kernel access the BIOS. The interface to the CBIOS from the kernel is provided by the Dos_Hlp (Dos Helper Services). These are not available for access by PDDs, VDDs or FSDs, however a limited set of Dos_Hlp calls are provided via the TESTCFG.SYS and OEMHLP$ device drivers.

Notes:

- OEMHLP$ is not a separately loaded module - it is resident within the OS2LDR module.

OS2LDR is responsible for loading the Kernel at system initialisation time. It does not get involved with the loading of Application Programs, PDDs, VDDs for FSDs during normal running - that function is performed by the System Loader component of the Kernel.
The OS/2 Kernel's Interfaces
The OS/2 Kernel’s Interfaces (steady state)
Miscellaneous System Control Block Reference

The following control blocks are described in this section:

- System Anchor Segment (SAS)
- Block Management Package (BMP)
- Record Management Package (RMP)

An overview of the Miscellaneous System Control Blocks follows:

--------------------------------------------------------

Miscellaneous System Diagrams

The following diagrams are illustrated:

- The System Anchor Segment

--------------------------------------------------------

The System Anchor Segment
System Anchor Segment (SAS) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The SAS is the common anchor for many system control blocks and control block chains.

**Pointers**

70:0 maps the SAS as a read-only segment.
78:0 maps the SAS as a read/write segment.

**Locations**

Built statically within the OS2KRNL load module.

**VM Owner**

os2krnl (0xffaa)

**Format**

**SAS Base Section.**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_signature</td>
<td>+0</td>
<td>4</td>
<td>A</td>
<td>&quot;SAS &quot;</td>
</tr>
<tr>
<td>SAS_tables_data</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>offset to tables section</td>
</tr>
<tr>
<td>SAS_flat_sel</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>FLAT selector for kernel data</td>
</tr>
<tr>
<td>SAS_config_data</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>offset to configuration section</td>
</tr>
<tr>
<td>SAS_dd_data</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>offset to device driver section</td>
</tr>
<tr>
<td>SAS_vm_data</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>offset to Virtual Memory section</td>
</tr>
<tr>
<td>SAS_task_data</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>offset to Tasking section</td>
</tr>
<tr>
<td>SAS_RAS_data</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>offset to RAS section</td>
</tr>
<tr>
<td>SAS_file_data</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>offset to File System section</td>
</tr>
<tr>
<td>SAS_info_data</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>offset to infoseg section</td>
</tr>
</tbody>
</table>

**SAS_tables_section** Protected Mode tables section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_tbl_GDT</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>selector for GDT</td>
</tr>
<tr>
<td>SAS_tbl_LDT</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>selector for LDT</td>
</tr>
<tr>
<td>SAS_tbl&gt;IDT</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>selector for IDT</td>
</tr>
<tr>
<td>SAS_tbl_GDTPOOL</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>selector for GDTPool</td>
</tr>
</tbody>
</table>

**SAS_config_section** Configuration Section section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_config_table</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>offset for Device Configuration Table</td>
</tr>
</tbody>
</table>
### SAS_dd_section Device Driver Section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_dd_bimodal_chain</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>offset for the first bimodal device driver's device header</td>
</tr>
<tr>
<td>SAS_dd_real_chain</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>offset for the address of the first real mode device driver's device header</td>
</tr>
<tr>
<td>SAS_dd_DPB_segment</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>selector for Drive Parameter Block (DPB) segment</td>
</tr>
<tr>
<td>SAS_dd_CDA_anchor_p</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>selector for ABIOS protected mode Common Data Area</td>
</tr>
<tr>
<td>SAS_dd_CDA_anchor_r</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>segment for ABIOS real mode Common Data Area</td>
</tr>
<tr>
<td>SAS_dd_FSC</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>selector for FSC</td>
</tr>
</tbody>
</table>

### SAS_vm_section Virtual Memory Management section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_vm_arena</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Flat offset of arena records</td>
</tr>
<tr>
<td>SAS_vm_object</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Flat offset of object records</td>
</tr>
<tr>
<td>SAS_vm_context</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Flat offset of context records</td>
</tr>
<tr>
<td>SAS_vm_krn1_mte</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Flat offset of kernel MTE records</td>
</tr>
<tr>
<td>SAS_vm_glbl_mte</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Flat offset of global MTE linked list. Note this field points into the chain to pick up global MTEs only. Use SAS_vm_all_mte to find all the MTEs.</td>
</tr>
<tr>
<td>SAS_vm_pft</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Flat offset of page frame table</td>
</tr>
<tr>
<td>SAS_vm_prt</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Flat offset of page range table</td>
</tr>
<tr>
<td>SAS_vm_swap</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Pointer to flat offset of swapper disk frame bit map followed by the size of the bit map in bits WARNING: the bit map offset and size are volatile</td>
</tr>
<tr>
<td>SAS_vm_idle_head</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Flat offset of Idle Head</td>
</tr>
<tr>
<td>SAS_vm_free_head</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Flat offset of Free Head</td>
</tr>
<tr>
<td>SAS_vm_heap_info</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Flat offset of Heap Array</td>
</tr>
<tr>
<td>SAS_vm_all_mte</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Flat offset of all MTEs linked list</td>
</tr>
</tbody>
</table>

### SAS_task_section Tasking section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_task_PTDA</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>selector for current PTDA</td>
</tr>
<tr>
<td>SAS_task_ptdaptrs</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>FLAT offset for process tree head</td>
</tr>
<tr>
<td>SAS_task_threadaptrs</td>
<td>+6</td>
<td>4</td>
<td>D</td>
<td>FLAT address for TCB address array</td>
</tr>
<tr>
<td>SAS_task_tasknumber</td>
<td>+a</td>
<td>4</td>
<td>D</td>
<td>offset for current TCB number</td>
</tr>
</tbody>
</table>
SAS_task_threadcount +e 4 D offset for ThreadCount

SAS_RAS_section RAS section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_RAS_STDA_p</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>selector for System Trace Data Area (STDA)</td>
</tr>
<tr>
<td>SAS_RAS_STDA_r</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>segment for System Trace Data Area (STDA)</td>
</tr>
<tr>
<td>SAS_RAS_event_mask</td>
<td>+6</td>
<td>4</td>
<td>D</td>
<td>offset for trace event mask</td>
</tr>
</tbody>
</table>

SAS_file_section File System section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_file_MFT</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>handle to MFT PTree</td>
</tr>
<tr>
<td>SAS_file_SFT</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>selector for System File Table (SFT) segment</td>
</tr>
<tr>
<td>SAS_file_VPB</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>selector for Volume Parameter Block (VPB) segment</td>
</tr>
<tr>
<td>SAS_file_CDS</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>selector for Current Directory Structure (CDS) segment</td>
</tr>
<tr>
<td>SAS_file_buffers</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>selector for buffer segment</td>
</tr>
</tbody>
</table>

SAS_info_section Information Segment section.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS_info_global</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>selector for global info seg</td>
</tr>
<tr>
<td>SAS_info_local</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>address of curtask local infoseg</td>
</tr>
<tr>
<td>SAS_info_localRM</td>
<td>+6</td>
<td>4</td>
<td>D</td>
<td>address of DOS task's infoseg</td>
</tr>
<tr>
<td>SAS_info_CDIB</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>selector for Codepage Data</td>
</tr>
</tbody>
</table>

Block Management Package Header (VMBH) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The BMP is a generalised facility used to manage tables of fixed length entities. The BMP consists of a header followed by the table it manages. The use of the BMP is many and varied, but almost always occurs where an expandable table of fixed length entries is required by the system.

Pointers

The VMBH prefixes tables, which are pointed to by:

_parVMOne  The table of VM Arena Records (VMARs).
The table of VM Object Records (VMOBs).
The table of VM Alias Records (VMALs).
The table of VM Context Records (VMCO).
BMP Selector for Device Driver Strategy 2 Request Packets.
ListIO BMP segment.
BMP Selector for Named Pipe NP Structures.
Volume Parameters Block BMP.
Floating point emulator BMP.
BMP for Swappable Kernel Heap Descriptors (VMKSHD).
BMP for Memory Lock Handles.
PerfView counters.
The array of PTDA structures.
The array of TCB structures.

Locations
Many.

VM Owner
The VMBH is usaualy part of the the object from which the BMP is allocted, thus adopts the object id of the table.

Format

VMBH BMP Header Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bh_pbFreeHead</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Free list pointer</td>
</tr>
<tr>
<td>bh_pbEndBlocks</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>End of valid blocks</td>
</tr>
<tr>
<td>bh_pbEndVirt</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>End of virtual memory</td>
</tr>
<tr>
<td>bh_pbLastBusy</td>
<td>+C</td>
<td>4</td>
<td>D</td>
<td>Pointer to last busy block</td>
</tr>
<tr>
<td>bh_pfnisbusy</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Busy block identifier function</td>
</tr>
<tr>
<td>bh_filler</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Paragraph boundary filler</td>
</tr>
<tr>
<td>bh_flpgtype</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>New page type flags</td>
</tr>
<tr>
<td>bh_cbPerBlock</td>
<td>+1c</td>
<td>2</td>
<td>W</td>
<td>Size of a block in bytes</td>
</tr>
<tr>
<td>bh_hob</td>
<td>+1e</td>
<td>2</td>
<td>W</td>
<td>Object record handle</td>
</tr>
</tbody>
</table>

bh_flpgtype flag definitions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG_CONTIG</td>
<td>0x00000001</td>
<td>contiguous physical memory</td>
</tr>
<tr>
<td>PG_NOINCR</td>
<td>0x00000001</td>
<td>don’t increment physical addr</td>
</tr>
<tr>
<td>PG_W</td>
<td>0x00000002</td>
<td>Writable - value from pte</td>
</tr>
<tr>
<td>PG_U</td>
<td>0x00000004</td>
<td>user mode accessible - from pte</td>
</tr>
<tr>
<td>PG_X</td>
<td>0x00000008</td>
<td>eXecutable</td>
</tr>
</tbody>
</table>
Record Management Package (RMP) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The RMP is used to manage tables of variable length entities. It appears in a number of situations, particularly those that required ASCII strings, such as file names, to be managed.

**Pointers**

rp_selector of the RMP handle maps the RMP segment.

**Locations**

RMP handles are located at the following labels:

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharDevRMPRec</td>
<td>Character Device Drivers</td>
</tr>
<tr>
<td>SpoolDevRMPRec</td>
<td>Spooler Device Drivers</td>
</tr>
<tr>
<td>NmpRmpHand</td>
<td>Named Pipes</td>
</tr>
<tr>
<td>hDiscSegRmpStruc</td>
<td>Discardable Segments</td>
</tr>
<tr>
<td>ShareRmpStruc</td>
<td>Named Shared Memory</td>
</tr>
<tr>
<td>SysSemRmpHdl</td>
<td>System Semaphores</td>
</tr>
</tbody>
</table>

**VM Owner**

<table>
<thead>
<tr>
<th>Label</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharDevRMPRec</td>
<td>chardevrmp (0xff35)</td>
</tr>
<tr>
<td>SpoolDevRMPRec</td>
<td>spldevrmp (0xff34)</td>
</tr>
<tr>
<td>NmpRmpHand</td>
<td>npipenpn (0xff30)</td>
</tr>
<tr>
<td>hDiscSegRmpStruc</td>
<td>discard (0xff6c)</td>
</tr>
<tr>
<td>ShareRmpStruc</td>
<td>mshrmp (0xff83)</td>
</tr>
<tr>
<td>SysSemRmpHdl</td>
<td>syssemmrmp (0xff36)</td>
</tr>
</tbody>
</table>

**Format**

rbhead RMP Header Structure.

**Field Name Offset Length Type Description**
**rbfree** RMP Free Record Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rf_size</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>free block size (high bit set)</td>
</tr>
<tr>
<td>rf_prev_free</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>link to prev free block in seg</td>
</tr>
<tr>
<td>rf_next_free</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>link to next free block in seg</td>
</tr>
</tbody>
</table>

**rparm** RMP Handle Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rp_flags</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>flags</td>
</tr>
<tr>
<td></td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>unused</td>
</tr>
<tr>
<td>rp_selector</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>GDT selector to use</td>
</tr>
</tbody>
</table>

**rp_flags** flag definitions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPF_BUSY</td>
<td>0x01</td>
<td>Segment busy flag</td>
</tr>
<tr>
<td>RPF_WAITING</td>
<td>0x02</td>
<td>Somebody waiting flag</td>
</tr>
<tr>
<td>RPF_ALLOC</td>
<td>0x04</td>
<td>Segment allocated flag</td>
</tr>
</tbody>
</table>

Semaphore Control Block Reference

The following control blocks are described in this section:

FastSafeRamSemStruc
FastSafeRamSemStruc PM version

MuxTableEntry
RamSemStruc
Kernel Semaphore Structures
32-bit Semaphore Structures
System Semaphore Structures
PM/GRE Semaphore

--------------------------------------------

FastSafeRamSemStruc

Pointers

TCB_SemInfo points to fs_RAMSem

Locations

Multiple, in user storage.

VM Owner

Multiple user storage owners.

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastSafeRamSemStruc</td>
<td>-a</td>
<td>e</td>
<td>S</td>
<td>Fast Safe Ram Semaphore</td>
</tr>
<tr>
<td>fs_Length</td>
<td>-a</td>
<td>2</td>
<td>W</td>
<td>Length of this structure</td>
</tr>
<tr>
<td>fs_ProcID</td>
<td>-8</td>
<td>2</td>
<td>W</td>
<td>Process ID of owner or zero</td>
</tr>
<tr>
<td>fs_ThrdID</td>
<td>-6</td>
<td>2</td>
<td>W</td>
<td>Thread ID of owner or zero</td>
</tr>
<tr>
<td>fs_Usage</td>
<td>-4</td>
<td>2</td>
<td>W</td>
<td>reference count</td>
</tr>
<tr>
<td>fs_Client</td>
<td>-2</td>
<td>2</td>
<td>W</td>
<td>16 bit field for use by owner</td>
</tr>
<tr>
<td>fs_RAMSem</td>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>OS/2 RAM Semaphore</td>
</tr>
</tbody>
</table>

--------------------------------------------

FastSafeRamSemStruc PM Version

Pointers

TCB_SemInfo points to fs_RAMSem

Locations

Multiple, in user storage.

VM Owner

Multiple user storage owners.

Format
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastSafeRamSemStruc</td>
<td>-e</td>
<td>12</td>
<td>S</td>
<td>PM Fast Safe Ram Semaphore</td>
</tr>
<tr>
<td>fs_Length</td>
<td>-e</td>
<td>2</td>
<td>W</td>
<td>Length of this structure</td>
</tr>
<tr>
<td>fs_ProcID</td>
<td>-c</td>
<td>2</td>
<td>W</td>
<td>Process ID of owner or zero</td>
</tr>
<tr>
<td>fs_ThrdID</td>
<td>-a</td>
<td>2</td>
<td>W</td>
<td>Thread ID of owner or zero</td>
</tr>
<tr>
<td>fs_Usage</td>
<td>-8</td>
<td>2</td>
<td>W</td>
<td>reference count</td>
</tr>
<tr>
<td>fs_Client</td>
<td>-6</td>
<td>2</td>
<td>W</td>
<td>16 bit field for use by owner</td>
</tr>
<tr>
<td>fs_Timeout</td>
<td>-4</td>
<td>4</td>
<td>D</td>
<td>Timeout value</td>
</tr>
<tr>
<td>fs_RAMSem</td>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>OS/2 RAM Semaphore</td>
</tr>
</tbody>
</table>

MuxTableEntry

Locations

At label MuxTable in system storage

VM Owner

os2krnl (0xffaa)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MuxTableEntry</td>
<td>+0</td>
<td>9</td>
<td>S</td>
<td>Mux Table Entry</td>
</tr>
<tr>
<td>MuxLink</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Selector Link to next entry. Used to chain entries for a MuxWait request</td>
</tr>
<tr>
<td>MuxThreadID</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread Slot ID of waiter</td>
</tr>
<tr>
<td>MuxType</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td>Semaphore type.</td>
</tr>
<tr>
<td>MuxSemID</td>
<td>+5</td>
<td>4</td>
<td>D</td>
<td>Mux Semaphore handle</td>
</tr>
</tbody>
</table>

MuxType flag definitions

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUXTYPE_CLEAR</td>
<td>0</td>
<td>the mux table entry is clear</td>
</tr>
<tr>
<td>MUXTYPE_SYSSEM</td>
<td>1</td>
<td>the ID is a system sem address</td>
</tr>
<tr>
<td>MUXTYPE_RAMHANDLE</td>
<td>2</td>
<td>the ID is a ram sem handle:offset</td>
</tr>
<tr>
<td>MUXTRYE_RAMPHY</td>
<td>3</td>
<td>the ID is a ram sem physical address</td>
</tr>
<tr>
<td>MUXTYPE_EVENTSEM</td>
<td>4</td>
<td>the ID for a 32-bit event sem</td>
</tr>
</tbody>
</table>

RamSemStruc
Pointers

TCB_SemInfo

Locations

Multiple, in user storage.

VM Owner

Multiple user storage owners.

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RamSemStruc</td>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>Ram Semaphore</td>
</tr>
<tr>
<td>RamSemOwner</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>Ownership flag</td>
</tr>
<tr>
<td>RamSemFlag</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>Ram Semaphore flag bit field</td>
</tr>
<tr>
<td>RamSemID</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>RamSem Block/Run ID low word</td>
</tr>
</tbody>
</table>

RamSemFlag definitions

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMSEM_WAITING</td>
<td>0x01</td>
<td>a thread is waiting on the sem</td>
</tr>
<tr>
<td>RAMSEM_MUXWAITING</td>
<td>0x02</td>
<td>a thread is muxwaiting on the sem</td>
</tr>
</tbody>
</table>

Notes:

The high-order 4 bit of the RamSemFlag are used as an extended owner field (to cater for more than 512 threads).

Only kernel code sets the RamSemOwner field to a thread slot number. Ring 3 RamSems have 0xff value for an owned RamSem

KSEM Structures for OS/2 Warp V4.0 and .OS/2 Warp V3.0 ALLSTRICT kernel

For KSEM formats for other versions of OS/2 see:

KSEM for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

Locations

Multiple, either imbeded in system structres, for example PTDA, MFT, or dynamically allocated from the kernel heaps.

VM Owner

Imbedded KSEMs assume the Owner Id of the imbedding structure. Stand-alone KSEMs allocated from the kernel heaps use id: ksem (0xff7e)

Format

KSEMSHR  Shared Kernel Semaphore
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ks_Signature</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ks_bFlags</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ks_bType</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ks_Owner</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusPendingWriters</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusNest</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusReaders</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusPendingReaders</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

**KSEM Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ksm_Signature</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksm_bFlags</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ksm_bType</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ksm_Owner</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksm_cusPendingWriters</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksm_cusNest</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kse_Signature</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>kse_bFlags</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>kse_bType</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>kse_Owner</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>kse_cusPendingWriters</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

**KSEM Flag Definitions.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSEM_NOINTERRUPT</td>
<td>0x1</td>
<td></td>
</tr>
<tr>
<td>KSEM_WRITER</td>
<td>0x2</td>
<td></td>
</tr>
<tr>
<td>KSEM_DISPLAYID</td>
<td>0x4</td>
<td></td>
</tr>
<tr>
<td>KSEM_NOBLOCK</td>
<td>0x8</td>
<td></td>
</tr>
</tbody>
</table>

---

**KSEM Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0**
RETAIL kernel

**KSEMSHR** Shared Kernel Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ks_bFlags</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ks_bType</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ks_Owner</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusPendingWriters</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusNest</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusReaders</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ks_cusPendingReaders</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

**KSEMMTX** MUTEX Kernel Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ksm_bFlags</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ksm_bType</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ksm_Owner</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksm_cusPendingWriters</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksm_cusNest</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

**KSEMEVT** Event Kernel Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kse_bFlags</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>kse_bType</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>kse_Owner</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>kse_cusPendingWriters</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

32-bit Semaphore Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRRICT kernel

For 32-bit Semaphore formats for other versions of OS/2 see:

32-bit Semaphore for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

**Pointers**

TCB_SleepId points to SEVENT, PEVENT, SMutex, PMutex, SMUX or PMUX when waiting on the semaphore.
PTDA field `pPrSemTbl` points to the private semaphore table, which is indexed by the semaphore handle.

`pShSemTbl` points to the shared semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structure.

PTDA field `pPrSemTbl` points to the per-process private semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structure.

`pShSemStrTbl` points to the table of `SEMTBLNODE` entries. Each of these points to a hashed chain of `SEMSTRNODE` structures.

**Note:** Names are hashed by treating each name as table of null padded ULONGs and successively adding.

**Locations**

Structures are allocated from the kernel heaps.

**VM Owners**

- `SEVENT` semstruc (0xffc2)
- `PEVENT` semstruc (0xfff2)
- `SMUTEX` semstruc (0xfff2)
- `PMUTEX` semstruc (0xfff2)
- `SMUX` semstruc (0xfff2)
- `PMUX` semstruc (0xfff2)
- `OPENQ` semopenq (0xffbf)
- `MUXQ` semmuxq (0xfffbe)
- `SEMRECORD` semrec (0xfff0)
- `SEMTBLNODE` semtable (0xfff3)
- `SEMSTRNODE` semtable (0xfff3)
- `Semaphore name` semstr (0xfff1)

**Format**

**SEVENT** Shared Event Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>pMuxQ</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>pointer to the mux queue</td>
</tr>
<tr>
<td>usPostCt</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>number of posts</td>
</tr>
<tr>
<td>pOpenQ</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>pointer to the open queue</td>
</tr>
<tr>
<td>pszName</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>name of semaphore, null if anonymous</td>
</tr>
<tr>
<td>pulCreatAddr</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Address passed in by app during create</td>
</tr>
<tr>
<td>ulSig</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>0x54564553 &quot;SEVT&quot;</td>
</tr>
<tr>
<td>ptcb</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>ptcb of caller</td>
</tr>
</tbody>
</table>

**PEVENT** Private Event Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Off</td>
<td>Length</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>--------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>pMuxQ</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>pointer to the mux queue</td>
</tr>
<tr>
<td>usPostCt</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>number of posts</td>
</tr>
<tr>
<td>pOpenCt</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>number of opens</td>
</tr>
<tr>
<td>pulCreateAddr</td>
<td>+a</td>
<td>4</td>
<td>D</td>
<td>Address passed in by app during create</td>
</tr>
<tr>
<td>ulSig</td>
<td>+e</td>
<td>4</td>
<td>D</td>
<td>0x54564550 &quot;PEVT&quot;</td>
</tr>
</tbody>
</table>

**SMUTEX** Shared Mutex Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>pMuxQ</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>pointer to the mux queue</td>
</tr>
<tr>
<td>usRequestCt</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>number of requests</td>
</tr>
<tr>
<td>usSlotNum</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>slot number of the owning thread</td>
</tr>
<tr>
<td>usRequesterCt</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>number of requesters</td>
</tr>
<tr>
<td>pOpenQ</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>pointer to the open queue</td>
</tr>
<tr>
<td>pszName</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>name of semaphore, null if anonymous</td>
</tr>
<tr>
<td>pulCreateAddr</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Address passed in by app during create</td>
</tr>
<tr>
<td>ulSig</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>0x58544D53 &quot;SMTX&quot;</td>
</tr>
</tbody>
</table>

**PMUTEX** Private Mutex Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>pMuxQ</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>pointer to the mux queue</td>
</tr>
<tr>
<td>usRequestCt</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>number of requests</td>
</tr>
<tr>
<td>usSlotNum</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>slot number of the owning thread</td>
</tr>
<tr>
<td>usRequesterCt</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>number of requesters</td>
</tr>
<tr>
<td>pOpenCt</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>number of opens</td>
</tr>
<tr>
<td>pulCreateAddr</td>
<td>+e</td>
<td>4</td>
<td>D</td>
<td>Address passed in by app during create</td>
</tr>
<tr>
<td>ulSig</td>
<td>+12</td>
<td>4</td>
<td>D</td>
<td>0x58544D50 &quot;PMTX&quot;</td>
</tr>
</tbody>
</table>

**SMUX** Shared Mux Wait Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>cSemRec</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>count of semaphore records</td>
</tr>
<tr>
<td>pSemRec</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>array of semaphore record entries</td>
</tr>
<tr>
<td>usWaitCt</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>number of threads waiting on the mux</td>
</tr>
<tr>
<td>pOpenQ</td>
<td>+a</td>
<td>4</td>
<td>D</td>
<td>pointer to the open queue</td>
</tr>
</tbody>
</table>
pszName            +e       2        W        name of semaphore, null if anonymous
pulCreatAddr       +10      4        D        Address passed in by app during create
ulSig              +14      4        D        0x58554D53 "SMUX"

**PMUX** Private Mux Wait Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>cSemRec</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>count of semaphore records</td>
</tr>
<tr>
<td>pSemRec</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>array of semaphore record entries</td>
</tr>
<tr>
<td>usWaitCt</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>number of threads waiting on the mux</td>
</tr>
<tr>
<td>usOpenCt</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>number of opens</td>
</tr>
<tr>
<td>pPTDA</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>pointer to PTDA of creator</td>
</tr>
<tr>
<td>pulCreatAddr</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Address passed in by app during create</td>
</tr>
<tr>
<td>ulSig</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>0x58554D50 &quot;PMUX&quot;</td>
</tr>
</tbody>
</table>

**OPENQ** Open Queue Node Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pidOpener</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>process id of opening process</td>
</tr>
<tr>
<td>usOpenCt</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>number of Opens for this process</td>
</tr>
<tr>
<td>pNextOpen</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to next node in list</td>
</tr>
<tr>
<td>ulSig</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>0x514E504F &quot;OPNQ&quot;</td>
</tr>
</tbody>
</table>

**MUXQ** Mux Queue Node Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pMux</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>pointer to a mux (shared or private)</td>
</tr>
<tr>
<td>pNextMux</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to next mux waiter in list</td>
</tr>
<tr>
<td>ulSig</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>0x5158554D &quot;MUXQ&quot;</td>
</tr>
</tbody>
</table>

**SEMRECORD** Semaphore Record Structure for MUX Wait Semaphores.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsemCur</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>semaphore handle</td>
</tr>
<tr>
<td>ulUser</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>user value</td>
</tr>
</tbody>
</table>

**SEMSTRNODE** Semaphore String Node

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsem</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>semaphore handle</td>
</tr>
</tbody>
</table>
psz                +4   4        D    pointer to the string
pNext              +8   4        D    pointer to next string node
ulSig              +c   4        D    0x444F4E53 "SNOD"

**SEMTBLNODE** Semaphore String Node Table Entry

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ulKey</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>hash key</td>
</tr>
<tr>
<td>pStrNode</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to string node</td>
</tr>
</tbody>
</table>

**usFlags** field definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE_POSTED</td>
<td>0x0040</td>
<td>The event sem APIs set this flag if the event is in the posted state</td>
</tr>
<tr>
<td>DM_OWNER_DIED</td>
<td>0x0080</td>
<td>The process died while owning the mutex semaphore</td>
</tr>
<tr>
<td>DMW_MTX_MUX</td>
<td>0x0100</td>
<td>The muxwait semaphore APIs set this flag if the mux contains mutex sems</td>
</tr>
<tr>
<td>DHO_SEM_OPEN</td>
<td>0x0200</td>
<td>dh_OpenEventSem sets this flag to indicate that device drivers have opened the given semaphore</td>
</tr>
<tr>
<td>DE_16BIT_MW</td>
<td>0x0400</td>
<td>Part of a 16-bit MuxWait if this flag is set</td>
</tr>
</tbody>
</table>

---------------------------------------------------------------

**32-bit Semaphore Structures for OS/2 Warp V3.0 RETAIL kernel**

**SEVENT** Shared Event Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usFlags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>attributes</td>
</tr>
<tr>
<td>pMuxQ</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>pointer to the mux queue</td>
</tr>
<tr>
<td>usPostCt</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>number of posts</td>
</tr>
<tr>
<td>pOpenQ</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>pointer to the open queue</td>
</tr>
<tr>
<td>pszName</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>name of semaphore, null if anonymous</td>
</tr>
<tr>
<td>ptcb</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>ptcb of caller</td>
</tr>
</tbody>
</table>

**PEVENT** Private Event Semaphore
**Field Name** | **Off** | **Length** | **Type** | **Description**
---|---|---|---|---
usFlags | +0 | 2 | W | attributes
pMuxQ | +2 | 4 | D | pointer to the mux queue
usPostCt | +6 | 2 | W | number of posts
pOpenCt | +8 | 2 | W | number of opens

**SMUTEX** Shared Mutex Semaphore

**Field Name** | **Off** | **Length** | **Type** | **Description**
---|---|---|---|---
usFlags | +0 | 2 | W | attributes
pMuxQ | +2 | 4 | D | pointer to the mux queue
usRequestCt | +6 | 2 | W | number of requests
usSlotNum | +8 | 2 | W | slot number of the owning thread
usRequesterCt | +a | 2 | W | number of requesters
pOpenQ | +c | 4 | D | pointer to the open queue
pszName | +10 | 4 | D | name of semaphore, null if anonymous

**PMUTEX** Private Mutex Semaphore

**Field Name** | **Off** | **Length** | **Type** | **Description**
---|---|---|---|---
usFlags | +0 | 2 | W | attributes
pMuxQ | +2 | 4 | D | pointer to the mux queue
usRequestCt | +6 | 2 | W | number of requests
usSlotNum | +8 | 2 | W | slot number of the owning thread
usRequesterCt | +a | 2 | W | number of requesters
usOpenCt | +c | 2 | W | number of opens

**SMUX** Shared Mux Wait Semaphore

**Field Name** | **Off** | **Length** | **Type** | **Description**
---|---|---|---|---
usFlags | +0 | 2 | W | attributes
cSemRec | +2 | 2 | W | count of semaphore records
pSemRec | +4 | 4 | D | array of semaphore record entries
usWaitCt | +8 | 2 | D | number of threads waiting on the mux
pOpenQ | +a | 4 | D | pointer to the open queue
pszName | +e | 2 | W | name of semaphore, null if anonymous

**PMUX** Private Mux Wait Semaphore

**Field Name** | **Off** | **Length** | **Type** | **Description**
---|---|---|---|---
usFlags | +0 | 2 | W | attributes
cSemRec            +2   2        W    count of semaphore records
pSemRec            +4   4        D    array of semaphore record entries
usWaitCt           +8   2        W    number of threads waiting on the mux
usOpenCt           +a   2        W    number of opens
pPTDA              +c   4        D    pointer to PTDA of creator

**OPENQ** Open Queue Node Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pidOpener</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>process id of opening process</td>
</tr>
<tr>
<td>usOpenCt</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>number of Opens for this process</td>
</tr>
<tr>
<td>pNextOpen</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to next node in list</td>
</tr>
</tbody>
</table>

**MUXQ** Mux Queue Node Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pMux</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>pointer to a mux (shared or private)</td>
</tr>
<tr>
<td>pNextMux</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to next mux waiter in list</td>
</tr>
</tbody>
</table>

**SEMSTRNODE** Semaphore String Node

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsem</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>semaphore handle</td>
</tr>
<tr>
<td>psz</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>pointer to the string</td>
</tr>
<tr>
<td>pNext</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>pointer to next string node</td>
</tr>
</tbody>
</table>

System Semaphore Structures

**Pointers**

SysSemRmpHdl contains the selector that points the system semaphore names RMP.

**Locations**

SysSemDataTable is the location of the global system semaphores table. Each entry is a SysSemTblStruc structure.

PTDA field SysSemPTDATbl is the location of the per-process semaphore table.

PTDA per-process semaphore contains byte-length entries, which are per-semaphore use counts.

The semaphore handle indexes both the per-process and global semaphore tables.

SysSemHighTable locates the table of SysSemHighTableS structures.
VM Owner

syserrmp (0xff36) for the RMP that contains the semaphore names.

Other global tables are owned by os2knl (0xfffa).

Format

SysSemHandleStruc System Semaphore Handle Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysSemHighWord</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>0x8000 for sys sems</td>
</tr>
<tr>
<td>SysSemPTDAIndex</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Index into the PTDA open sem table</td>
</tr>
</tbody>
</table>

SysSemTblStruc System Semaphore Table Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysSemOwner</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>thread owning this semaphore</td>
</tr>
<tr>
<td>SysSemFlag</td>
<td>+2</td>
<td>1</td>
<td>B</td>
<td>system semaphore flag bit field</td>
</tr>
<tr>
<td>SysSemRefCount</td>
<td>+3</td>
<td>1</td>
<td>B</td>
<td>number of references to this sys sem</td>
</tr>
<tr>
<td>SysSemProcCnt</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td>number of requests for this owner</td>
</tr>
<tr>
<td>SysSemPad</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td>pad byte to round structure up to word</td>
</tr>
</tbody>
</table>

SysSemHighTableS System Semaphore Table Extension Structure.

This is an extension of the SysSemTblStruc that is put into high memory so we don't impact the low data segment. It is only used in protected mode during process/thread termination.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysSemPidOwner</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>pid owner, the thread owner has died</td>
</tr>
</tbody>
</table>

SysSemNameStruc System Semaphore Name table structure, managed by an RMP.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysSemPtr</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

SysSemFlag flag field definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSEMM_WAITING</td>
<td>0x01</td>
<td>a thread is waiting on the sem</td>
</tr>
<tr>
<td>SYSEMM_MUXWAITING</td>
<td>0x02</td>
<td>a thread is muxwaiting on the sem</td>
</tr>
<tr>
<td>SYSEMM_OWNER_DIED</td>
<td>0x04</td>
<td>the process/thread owning the sem died</td>
</tr>
<tr>
<td>SYSEMM_EXCLUSIVE</td>
<td>0x08</td>
<td>indicates a exclusive system semaphore</td>
</tr>
<tr>
<td>SYSEMM_NAME_CLEANUP</td>
<td>0x10</td>
<td>name table entry needs to be removed</td>
</tr>
<tr>
<td>SYSEMM_THREAD_OWNER_DIED</td>
<td>0x20</td>
<td>the thread owning the sem died</td>
</tr>
<tr>
<td>SYSEMM_EXITLIST_OWNER</td>
<td>0x40</td>
<td>the exitlist thread owns the sem</td>
</tr>
</tbody>
</table>
PM/GRE Semaphore Structure

Locations

`pmsemaphores` locates the table of PM/GRE semaphores.

VM Owner

PMMERGE.DLL `hmte`

Format

**GRESEM** PM/GRE Semaphore

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acIdent</td>
<td>+0</td>
<td>7</td>
<td>A</td>
<td>GRESEM or PMSEM</td>
</tr>
<tr>
<td>fcSet</td>
<td>+7</td>
<td>1</td>
<td>B</td>
<td>386 Actual Semaphore</td>
</tr>
<tr>
<td>ulProcessThread</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>owner process and thread id (PTid)</td>
</tr>
<tr>
<td>ulNestedUseCount</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td># of times same PTid has accessed sem</td>
</tr>
<tr>
<td>ulWaitingCount</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td># of PTids waiting on semaphore</td>
</tr>
<tr>
<td>ulUseCount</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td># of times semaphore has been used</td>
</tr>
<tr>
<td>ulEventHandle</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Event Handle Semaphore</td>
</tr>
<tr>
<td>ulCallerAddr</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Semaphore Caller</td>
</tr>
</tbody>
</table>

Memory Management Control Block Reference

The following control blocks are described in this section:

- Page Frame Structure (PF)
- Physical Arena Information Structures (PAI)
- Per Arena Page Table Data (PGDATA)
- Memory Alias Record (VMAL)
- Memory Arena Header (VMAH)
- Memory Arena Record (VMAR)
- Memory Arena Type (VMAT)
- Memory Context Record (VMCO)
- Memory Object Record (VMOB)
- Virtual Page Structure (VP)
- Kernel Heap Header (VMKH)
Kernel Resident Heap Structures (VMKRH, VMKRHY, VMKRHS, VMKRHF, VMKRHB, VMKRHBA)

Kernel Swappable Heap Structures (VMKSH, VMKSHD, VMKSHB)

An overview of the Memory Management Control Blocks follows:

--------------------------------------------

Memory Management Control Block Diagrams

The following diagrams illustrate the relationships between various Memory Management control blocks:

- Virtual Address Space Regions (OS/2 Warp V4.0)
- Virtual Address Space Regions (OS/2 Warp V3.0)
- Virtual Address Space Regions (OS/2 V2.11)
- Virtual Address Space Management
- Private Arena Private Data
- Private Arena Shared Data
- Shared Global Data
- Shared Arena Instance Data
- Virtual/Physical Page Management - Backed Storage
- Virtual/Physical Page Management - Swapped Storage
- CS Alias of Shared Instance Data
- Memory Alias in Multiple Processes

--------------------------------------------

Virtual Address Space Regions for OS/2 Warp V4.0 and OS/2 Warp V3.0 from fax pack 19
Virtual Address Space Regions
(OS/2 4.0 and 3.0 Fix Pack 19)

- System Arena
  - 4G-256K
  - 1.5G
  - 512M-64K
  - 448M
  - 416M
  - 384M
  - 320M
  - 304M
  - Expansion Region

- Reserved Regions
- Shared Arena
  - non-based code
  - Global/Shared Region
- Private Arena
  - process 1
  - process 2
  - process 3 (VDM)
Virtual Address Space Regions for OS/2 Warp V3.0
Virtual Address Space Regions for OS/2 V2.11
Virtual Address Space Regions
(OS/2 2.x)

System Arena

Shared Arena

Expansion Region

Private Arena
process 1

Private Arena
process 2

Private Arena
process 3
(VDM)
Virtual Address Space Management
Private Arena Shared Data
Private Arena Shared Data

Private Arena Process 1
- shared object

Private Arena Process 2
- shared object

physical mapping

physical storage (not copy)

VMAR

VMAR

link

hptea

hptea

VMOB

VMOB

va
	pseudo-object

va

VMOB

PTDA

VMOB

va

(pseudo-object)

MTE

VMOB

own

limit

pseudo-object

va
Shared Global Data
Shared Arena Instance Data
Virtual/Physical Page Management - Backed Storage
Page Management

Backed Virtual Storage

physical storage -> PF table -> VP table -> VMOB

VMAR

virtual storage

1K page

4K frame

PF

PTE

swap

physical mapping

module
Virtual/Physical Page Management - Swapped Storage
Page Management

Unbacked Virtual Storage

physical storage  →  PF table  →  VP table  →  YMOB  
(physical frame)  →  PF  →  VP  

virtual storage  →  4K page  

PTE  →  swapper  

physical mapping  →  module
CS Alias of Shared Instance Data
CS Alias of Shared Instance Data

Instance Data

VMAR

VMOB

VMOB

VMOB

CS Alias

VMAR

VMAL

DS

CS

Physical mapping

Data Object

Code Object

Physical Storage
Memory Alias in Multiple Processes
Memory Aliases in Multiple Processes
Page Frame Structure

Pointers

_.pft_ points to the table of Page Frame Structures.

Locations

System Arena

VM Owner

pgpf (0xffb4)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apf_s</td>
<td>+0</td>
<td>c</td>
<td>S</td>
<td>active pf</td>
</tr>
<tr>
<td>pf_pvp</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>vp cross link</td>
</tr>
<tr>
<td>pf_lock</td>
<td>+4</td>
<td>1</td>
<td>D</td>
<td>count of long term locks</td>
</tr>
<tr>
<td>pf_flags</td>
<td>0.4</td>
<td></td>
<td></td>
<td>flags</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td>pad</td>
</tr>
<tr>
<td>pf_refcount</td>
<td>2</td>
<td></td>
<td></td>
<td>count of ptes marked present</td>
</tr>
<tr>
<td>pf_block</td>
<td>+8</td>
<td>2.4</td>
<td>D</td>
<td>swp disk frame or ldr block number</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td>pad</td>
</tr>
<tr>
<td>pf_slock</td>
<td>1</td>
<td></td>
<td></td>
<td>count of short term locks</td>
</tr>
<tr>
<td>ipf_s</td>
<td>+0</td>
<td>c</td>
<td>S</td>
<td>idle page frame</td>
</tr>
<tr>
<td>pf_pvp</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>vp cross link</td>
</tr>
<tr>
<td>pf_flink1</td>
<td>+4</td>
<td>1</td>
<td>D</td>
<td>forward link part 1 (low byte)</td>
</tr>
<tr>
<td>pf_flags</td>
<td>0.4</td>
<td></td>
<td></td>
<td>flags</td>
</tr>
<tr>
<td>vp_blink</td>
<td>2.4</td>
<td></td>
<td></td>
<td>backward link</td>
</tr>
<tr>
<td>pf_block</td>
<td>+8</td>
<td>2.4</td>
<td>D</td>
<td>swp disk frame or ldr block number</td>
</tr>
<tr>
<td>pf_flink2</td>
<td>1.4</td>
<td></td>
<td></td>
<td>forward link part 2 (high 1.4 bytes)</td>
</tr>
<tr>
<td>fpf_s</td>
<td>+0</td>
<td>c</td>
<td>S</td>
<td>free page frame</td>
</tr>
<tr>
<td></td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>pad</td>
</tr>
<tr>
<td>pf_flink1</td>
<td>+4</td>
<td>1</td>
<td>D</td>
<td>forward link part 1 (low byte)</td>
</tr>
<tr>
<td>pf_flags</td>
<td>0.4</td>
<td></td>
<td></td>
<td>flags</td>
</tr>
<tr>
<td>vp_blink</td>
<td>2.4</td>
<td></td>
<td></td>
<td>backward link</td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>2.4</td>
<td>D</td>
<td>pad</td>
</tr>
<tr>
<td>pf_flink2</td>
<td>1.4</td>
<td></td>
<td></td>
<td>forward link part 2 (high 1.4 bytes)</td>
</tr>
</tbody>
</table>

_pf_flag_ flag definitions:
<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF_FAST</td>
<td>0x1</td>
<td>frame is fast memory</td>
</tr>
<tr>
<td>PF_BUSY</td>
<td>0x2</td>
<td>frame is busy</td>
</tr>
<tr>
<td>PF_FREE</td>
<td>0x4</td>
<td>frame is free</td>
</tr>
<tr>
<td>PF_RES</td>
<td>0x8</td>
<td>reserved</td>
</tr>
</tbody>
</table>

Physical Arena Information Structures

Pointers

SAS_vm_prt

Locations

System Arena.

Two PAIs exist as part of the kernel load module. These are located at labels _pgPageablePAI and _pgResidentPAI.

VM Owners

os2krnl (0xffaa)

Format

PAI Physical Arena Information Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pai_pprt</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>pointer to page range table</td>
</tr>
<tr>
<td>pai_nranges</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>number of ranges in range table</td>
</tr>
<tr>
<td>pai_1M</td>
<td>+8</td>
<td>c</td>
<td>S</td>
<td>1M boundary structure</td>
</tr>
<tr>
<td>pb_frame</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>last frame before boundary</td>
</tr>
<tr>
<td>pb_ppf</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>last pf before boundary</td>
</tr>
<tr>
<td>pb_ppr</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>page range containing boundary</td>
</tr>
<tr>
<td>pai_16M</td>
<td>+14</td>
<td>c</td>
<td>S</td>
<td>16M boundary structure</td>
</tr>
<tr>
<td>pb_frame</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>last frame before boundary</td>
</tr>
<tr>
<td>pb_ppf</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>last pf before boundary</td>
</tr>
<tr>
<td>pb_ppr</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>page range containing boundary</td>
</tr>
<tr>
<td>pai_end</td>
<td>+20</td>
<td>c</td>
<td>S</td>
<td>end of memory boundary structure</td>
</tr>
<tr>
<td>pb_frame</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>last frame before boundary</td>
</tr>
<tr>
<td>pb_ppf</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>last pf before boundary</td>
</tr>
<tr>
<td>pb_ppr</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>page range containing boundary</td>
</tr>
</tbody>
</table>
pagerange_s Page Range Table Entry.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pri_lastframe</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>last valid page in range</td>
</tr>
<tr>
<td>pri_firstframe</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>first valid page in range</td>
</tr>
</tbody>
</table>

Per Arena Page Table Data for OS/2 Warp V4.0 and OS/2 Warp V3.0

For **PGDATA** formats for other versions of OS/2 see:

**PGDATA** for OS/2 Warp 3.0 SMP

**Pointers**

For private arenas, **ptda_ppgdata**.

**Locations**

System Arena.

For private data, a PGDATA structure is imbedded in each process’ **PTDA** at *(ptda_pgdatal +0x80)*.

For the Global Shared Region the PGDATA structure is located at public symbol **_pgShrData**.

For the System Arena the PGDATA structure is located at public symbol **_pgData**.

The tables of PTE counts pointed to by **pd_pcalloc**, **pd_pcpresent** and **pd_pcresident** are located as follows:

**System arena:**

- **pd_pcalloc**
  Located at public symbol **_pgcPteAllocated**

- **pd_pcpresent**
  Located at public symbol **_pgcPtePresent**

- **pd_pcresident**
  Located at public symbol **_pgcPteResident**

**Global System Region:**

- **pd_pcalloc**
  Located at public symbol **_pgcShrPteAllocated**

- **pd_pcpresent**
  Located at public symbol **_pgcShrPtePresent**

- **pd_pcresident**
  Located at public symbol **_pgcShrPteResident**

**Private Data:**

All three tables are imbedded contiguously within the **PTDA** at **ptda_pgc**.

**Note:** The tables pointed to by **pd_ppde**, **pd_ppte**, **pd_pcalloc**, **pd_pcpresent** and **pd_pcresident** are indexed relative to the base virtual page number (**pd_base**) of the region.

**VM Owners.**

- **ptda** *(0xffcb)* **os2krnl** *(0xffaa)*

**Format**

**PGDATA** Per Arena Page Table Data.
### Field Name | Off | Length | Type Description
--- | --- | --- | ---
`pd_ppte` | +0 | 4 | `D` pointer to ptes
`pd_ppde` | +4 | 4 | `D` pointer to pdes
`pd_pcallo` | +8 | 4 | `D` pointer to (word) counts of allocated ptes per pde
`pd_pcpresent` | +c | 4 | `D` pointer to (word) counts of present ptes per pde
`pd_president` | +10 | 4 | `D` pointer to (word) counts of locked/resident ptes
`pd_base` | +14 | 4 | `D` base virtual page number
`pd_pvdma` | +18 | 4 | `D` base of vdm alias region
`pd_maxpde` | +1c | 2 | `W` max potential pdes for this arena
`pd_cpdelow` | +1e | 2 | `W` count of low in-use pdes
`pd_cpdem` | +20 | 2 | `W` count of high in-use pdes
`pd_ptcontext` | +22 | 2 | `W` page table context
`pd_flags` | +24 | 2 | `W` per-process page manager flags

#### `pd_flags` flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD_AGINGNEEDED</td>
<td>1</td>
<td>process not aged in this sweep</td>
</tr>
<tr>
<td>PD_FREE</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

-----------------------------

**Per Arena Page Table Data for OS/2 Warp 3.0 SMP**

PGDATA Per Arena Page Table Data.

### Field Name | Off | Length | Type Description
--- | --- | --- | ---
`pd_ppte` | +0 | 4 | `D` pointer to ptes
`pd_ppde` | +4 | 4 | `D` pointer to pdes
`pd_pcallo` | +8 | 4 | `D` pointer to (word) counts of allocated ptes per pde
`pd_pcpresent` | +c | 4 | `D` pointer to (word) counts of present ptes per pde
`pd_president` | +10 | 4 | `D` pointer to (word) counts of locked/resident ptes
`pd_base` | +14 | 4 | `D` base virtual page number
`pd_pvdma` | +18 | 4 | `D` base of vdm alias region
`pd_cpdelow` | +1c | 2 | `W` count of low in-use pdes
`pd_cpdem` | +1e | 2 | `W` count of high in-use pdes
VM Arena Header

Locations

_ahvmSys locates the System Arena VMAH.

_ahvmShr locates the Shared Arena VMAH.

PTDA field ptda_ah locates each Private Arena VMAH.

VM Owner

For shared and system arenas: os2krnl (0xffaa)

For private arenas: ptda (0xffcb)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ah_pahNext</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Link to next arena</td>
</tr>
<tr>
<td>ah_pahPrev</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Link to previous arena</td>
</tr>
<tr>
<td>ah_parSen</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Handle of arena sentinel</td>
</tr>
<tr>
<td>ah_parFree</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Hint of 1st free block in arena</td>
</tr>
<tr>
<td>ah_papbm</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to bitmap directory</td>
</tr>
<tr>
<td>ah_paharHash</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Hash table pointer</td>
</tr>
<tr>
<td>ah_pat</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to per-type info</td>
</tr>
<tr>
<td>ah_fl</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Flags</td>
</tr>
<tr>
<td>ah_laddrMin</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Minimum address currently mapped</td>
</tr>
<tr>
<td>ah_laddrMax</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Max address currently mapped</td>
</tr>
<tr>
<td>ah_car</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Count of arena entries</td>
</tr>
<tr>
<td>ah_carBitmap</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Max entry count to need bitmap</td>
</tr>
<tr>
<td>ah_lbnNumMax</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Max bitmap number</td>
</tr>
<tr>
<td>ah_lbmeNumMax</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Max bitmap entry number</td>
</tr>
<tr>
<td>ah_lHashNumMax</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>Max hash table index</td>
</tr>
<tr>
<td>ah_hob</td>
<td>+3c</td>
<td>2</td>
<td>W</td>
<td>Arena header pseudo-handle</td>
</tr>
<tr>
<td>ah_filler</td>
<td>+3e</td>
<td>2</td>
<td>W</td>
<td>Make structure 4-byte multiple</td>
</tr>
</tbody>
</table>

ah_fl flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMAH_BITMAP_BYPASS</td>
<td>0x00000001</td>
<td>Worth bypassing bitmap</td>
</tr>
</tbody>
</table>
VM Alias Record

Pointers

_palVMAliases points to the VMAL table.

VM Owner

vmal (0xffe2)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmal</td>
<td>+0</td>
<td>8</td>
<td>S</td>
<td>VM alias record</td>
</tr>
<tr>
<td>al_har</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>handle to alias' arena record</td>
</tr>
<tr>
<td>al_hobptda</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>context the alias is created from</td>
</tr>
<tr>
<td>al_poff</td>
<td>+4</td>
<td>2.4</td>
<td>D</td>
<td>page offset of the alias from start of object</td>
</tr>
<tr>
<td>al_f</td>
<td></td>
<td>1.4</td>
<td></td>
<td>flags indicating type of alias</td>
</tr>
<tr>
<td>vmsal</td>
<td>+0</td>
<td>8</td>
<td>S</td>
<td>SEL alias record</td>
</tr>
<tr>
<td>sal_har</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>handle to alias' arena record</td>
</tr>
<tr>
<td>sal_selcode</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>code selector if cs alias</td>
</tr>
<tr>
<td>al_hobptda</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>context the alias is created from if MEMMAP alias</td>
</tr>
<tr>
<td>sal_cref</td>
<td>+4</td>
<td>1.2</td>
<td>D</td>
<td>reference count</td>
</tr>
<tr>
<td>sal_f</td>
<td></td>
<td>0.6</td>
<td></td>
<td>flags</td>
</tr>
<tr>
<td>sal_seldata</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>data selector if cs alias (unused for MEMMAP)</td>
</tr>
</tbody>
</table>

al_f flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL_ISBUSY</td>
<td>0x1</td>
<td>Set if record is busy</td>
</tr>
<tr>
<td>AL_CSALIAS</td>
<td>0x2</td>
<td>Set if cs alias record</td>
</tr>
<tr>
<td>AL_MEMMAP</td>
<td>0x4</td>
<td>Set if MemMapAlias record</td>
</tr>
<tr>
<td>ALDBGALIAS</td>
<td>0x8</td>
<td>Set if debug alias</td>
</tr>
<tr>
<td>AL_CSDSVALID</td>
<td>0x10</td>
<td>Set if ds selector valid</td>
</tr>
<tr>
<td>AL_DEVHLP</td>
<td>0x20</td>
<td>Set if Devhlp alias</td>
</tr>
<tr>
<td>AL_PRIV</td>
<td>0x40</td>
<td>Set if privatized alias</td>
</tr>
</tbody>
</table>
AL_VDM       0x80     Set if VDM alias
AL_NOALIAS   0x100    Set if UVIRT mapping in VDMs

**sal_f flag definitions:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAL_CSALIAS</td>
<td>AL_CSALIAS</td>
<td></td>
</tr>
<tr>
<td>SAL_MEMMAPALIAS</td>
<td>AL_MEMMAP</td>
<td></td>
</tr>
<tr>
<td>SAL_CSDSVALID</td>
<td>0x10</td>
<td>mustn't coincide with other alias types</td>
</tr>
<tr>
<td>SAL_ALIASREFSHIFT</td>
<td>0x6</td>
<td>Low six bits reserved for flags</td>
</tr>
<tr>
<td>SAL_ALIASREFMASK</td>
<td>0x0ffc0</td>
<td>reference count bits mask</td>
</tr>
</tbody>
</table>

--------------------------------------------

**VM Arena Record**

**Pointers**

_parvmOne points to the VMAR table.

**VM Owner**

vmar (0xffe3)

**Format**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmar_reg</td>
<td>+0</td>
<td>16</td>
<td>S</td>
<td>Regular Arena Record</td>
</tr>
<tr>
<td>ar_xf</td>
<td>+0</td>
<td>1.4</td>
<td>D</td>
<td>Extra flags</td>
</tr>
<tr>
<td>ar_cpg</td>
<td>2.4</td>
<td></td>
<td></td>
<td>Size in pages</td>
</tr>
<tr>
<td>ar_ipg</td>
<td>+4</td>
<td>2.4</td>
<td>D</td>
<td>Virtual page no.</td>
</tr>
<tr>
<td>ar_f</td>
<td>1.4</td>
<td></td>
<td></td>
<td>Flags</td>
</tr>
<tr>
<td>ar_harnext</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>Handle of next Arena Record</td>
</tr>
<tr>
<td>ar_harprev</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>Handle of previous Arena Record</td>
</tr>
<tr>
<td>ar_harlink</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Handle of associated Arena Record</td>
</tr>
<tr>
<td>ar_hashhash</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>Hash table link</td>
</tr>
<tr>
<td>ar_hob</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>Handle of Object Record</td>
</tr>
<tr>
<td>ar_hco</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>Context record handle (shar+shr data)</td>
</tr>
<tr>
<td>ar_hobptda</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>PTDA handle or NULL (prvar or shar + instance data)</td>
</tr>
<tr>
<td>ar_sel</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>Selector (sysarena only)</td>
</tr>
<tr>
<td>ar_hal</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>Alias record handle, ° ° °° °° means not an alias</td>
</tr>
</tbody>
</table>

--------------------------------------------
vmar_sen +0  16  S  Sentinel Arena Record
ar_xf  +0  1.4  D  Extra flags
ar_cpg  +1.4  2.4  Size in pages
ar_ipg  +4  2.4  D  Virtual page no.
ar_f  +6.4  1.4  Flags
ar_harnext +8  2  W  Handle of next Arena Record
ar_harprev +a  2  W  Handle of previous Arena Record
ar_harlink +c  2  W  Handle of associated Arena Record
ar_harhash +e  2  W  Hash table link
ar_ipgmax +10  4  D  Maximum lage no. in the arena
ar_unused +14  2  W  reserved

ar_f flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_INUSE</td>
<td>0x001</td>
<td>Record not on free list</td>
</tr>
<tr>
<td>AR_TAG</td>
<td>0x006</td>
<td>Record type mask</td>
</tr>
<tr>
<td>AR_TAGREG</td>
<td>0x000</td>
<td>Regular record</td>
</tr>
<tr>
<td>AR_TAGSEN</td>
<td>0x002</td>
<td>Sentinel</td>
</tr>
<tr>
<td>AR_TAGBSEN</td>
<td>0x006</td>
<td>Boundary sentinel</td>
</tr>
<tr>
<td>AR_SELMAP</td>
<td>0x008</td>
<td>Memory mapped by selector</td>
</tr>
<tr>
<td>AR_SELBASEALL</td>
<td>0x00c</td>
<td>Base selector map all</td>
</tr>
<tr>
<td>AR_SELMASK</td>
<td>0x00c</td>
<td>Selector map mask</td>
</tr>
<tr>
<td>AR_RELOAD</td>
<td>0x010</td>
<td>Pre-reserved for huge item or</td>
</tr>
<tr>
<td>AR_WRITE</td>
<td>0x020</td>
<td>Write permission</td>
</tr>
<tr>
<td>AR_USER</td>
<td>0x040</td>
<td>User pages</td>
</tr>
<tr>
<td>AR_EXEC</td>
<td>0x080</td>
<td>Executable Pages</td>
</tr>
<tr>
<td>AR_READ</td>
<td>0x100</td>
<td>Read permission</td>
</tr>
<tr>
<td>AR_HCO</td>
<td>0x200</td>
<td>Record linked to Context List</td>
</tr>
<tr>
<td>AR_GUARD</td>
<td>0x400</td>
<td>Guard pages</td>
</tr>
<tr>
<td>AR_SGS</td>
<td>0x800</td>
<td>Registered under Screen Group Switch</td>
</tr>
</tbody>
</table>

ar_xf flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_HCOH</td>
<td>0x001</td>
<td>context record handle &gt; 64k</td>
</tr>
</tbody>
</table>

--------------------------------------------

VM Arena Type Information Record
Points

VMAH field ah\_pat points to the associated VMAT.

Locations

_atvm locates the table of VMATs.

VM Owner

os2krnl (0xffaa)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>at_laddrInitMin</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Initial minimum</td>
</tr>
<tr>
<td>at_laddrInitMax</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Initial maximum</td>
</tr>
<tr>
<td>at_laddrAbsMin</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Abs minimum boundary</td>
</tr>
<tr>
<td>at_laddrAbsMax</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Abs minimum boundary</td>
</tr>
<tr>
<td>at_cbInitBetween</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Spacer between arenas</td>
</tr>
<tr>
<td>at_lHashNumbMask</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Hash number mask</td>
</tr>
<tr>
<td>at_lHashNumbShift</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Hash number shift</td>
</tr>
<tr>
<td>at_lHashNumbAbsMax</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Max hash table index</td>
</tr>
<tr>
<td>at_lHashMinSize</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Min hash table size</td>
</tr>
<tr>
<td>at_lbmNumbMask</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Bitmap number mask</td>
</tr>
<tr>
<td>at_lbmNumbShift</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Bitmap number shift</td>
</tr>
<tr>
<td>at_lbmNumbAbsMax</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Abs Max bitmap #</td>
</tr>
<tr>
<td>at_lbcbMinSize</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Min bitmap dir size</td>
</tr>
<tr>
<td>at_lbmNumbMask</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Bitmap entry # mask</td>
</tr>
<tr>
<td>at_lbmeNumbShift</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>Bitmap entry # shift</td>
</tr>
<tr>
<td>at_lbmeNumbAbsMax</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>Abs Max bitmap entry</td>
</tr>
<tr>
<td>at_lbmeBitNumbMask</td>
<td>+40</td>
<td>4</td>
<td>D</td>
<td>Bit number mask</td>
</tr>
<tr>
<td>at_lbmeBitNumbShift</td>
<td>+44</td>
<td>4</td>
<td>D</td>
<td>Bit number shift</td>
</tr>
<tr>
<td>at_flInit</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>Initial flags</td>
</tr>
<tr>
<td>at_Gran</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>Granularity</td>
</tr>
<tr>
<td>at_laddrMinNoWrap</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>Min no-hash wrap laddr</td>
</tr>
<tr>
<td>at_laddrMaxNoWrap</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>Max no-hash wrap laddr</td>
</tr>
<tr>
<td>at_harParent</td>
<td>+58</td>
<td>2</td>
<td>W</td>
<td>Parent arena</td>
</tr>
</tbody>
</table>

_atflInit flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMAT_PRIV_TILED</td>
<td>0</td>
</tr>
<tr>
<td>VMAT_PRIV_VDM</td>
<td>1</td>
</tr>
<tr>
<td>VMAT_SHR_TILED</td>
<td>2</td>
</tr>
<tr>
<td>VMAT_SYS</td>
<td>3</td>
</tr>
</tbody>
</table>
VM Context Record

Pointers

_pcovmOne points to the table of VMCOs.

Locations

System Arena.

VM Owner

vmco (0xffe5)

Format

Field Name Offset Length Type Description
co_hconext +0 2 W Index of next Context Record
co_hobptda +2 2 W PTDA handle
co_fb +4 1 B Context record flags

co_fb flag definitions:

Name   Bit Mask Description
CO_CREATOR 0x01 originating context
CO_PRIV 0x80 Privatized context
CO_HCOH 0x20 Next context record handle > 64k
CO_WRITE 0x02 Write permission
CO_USER 0x04 User storage
CO_EXEC 0x08 Executable
CO_READ 0x10 Read permission
CO_GUARD 0x40 Guard page

VM Object Record

Pointers

_pobvmOne points to the table of VMOBs.

Locations

System Arena.
VM Owner

```text
vmob (0xfff1)
```

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ob_har</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Arena Record handle</td>
</tr>
<tr>
<td>ob_hobnext</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Associated Object Record handle</td>
</tr>
<tr>
<td>ob_va</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Pseudo-object's virtual address</td>
</tr>
<tr>
<td>ob_fs</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Flags</td>
</tr>
<tr>
<td>ob_hobowner</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Owner i.d.</td>
</tr>
<tr>
<td>ob_hobmte</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>MTE handle</td>
</tr>
<tr>
<td>ob_wsemowner</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>I.d. of thread owning semaphore</td>
</tr>
<tr>
<td>ob_bsemcnt</td>
<td>+c</td>
<td>1</td>
<td>B</td>
<td>Counter and waiting flag</td>
</tr>
<tr>
<td>ob_clclock</td>
<td>+d</td>
<td>1</td>
<td>B</td>
<td>Count of all long-term locks</td>
</tr>
<tr>
<td>ob_cslock</td>
<td>+e</td>
<td>1</td>
<td>B</td>
<td>Count of all short-term locks</td>
</tr>
<tr>
<td>ob_xflags</td>
<td>+f</td>
<td>1</td>
<td>B</td>
<td>Extra flags</td>
</tr>
</tbody>
</table>

Note:

A complete list of system owner ids may be found under VM System Object Owner Ids in the Reference Tables section of the System Reference.

**ob_fs flag definitions:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB_PSEUDO</td>
<td>0x8000</td>
<td>Pseudo-object</td>
</tr>
<tr>
<td>OB_API</td>
<td>0x4000</td>
<td>API allocated object</td>
</tr>
<tr>
<td>OB_LOCKWAIT</td>
<td>0x2000</td>
<td>Some thread to wake in VMUnlock</td>
</tr>
<tr>
<td>OB_LALIAS</td>
<td>0x1000</td>
<td>Object has aliases</td>
</tr>
<tr>
<td>OB_SHARED</td>
<td>0x0800</td>
<td>Object’s contents are shared</td>
</tr>
<tr>
<td>OB_UVIRT</td>
<td>0x0400</td>
<td>UVirt object</td>
</tr>
<tr>
<td>OB_ZEROINIT</td>
<td>0x0200</td>
<td>Object is zero-initialized</td>
</tr>
<tr>
<td>OB_RESIDENT</td>
<td>0x0100</td>
<td>Initial allocation was resident</td>
</tr>
<tr>
<td>OB_LWMMEM</td>
<td>0x0040</td>
<td>Object is in low memory</td>
</tr>
<tr>
<td>OB_GUARD</td>
<td>0x0080</td>
<td>Page attribute/permission flags</td>
</tr>
<tr>
<td>OB_EXEC</td>
<td>0x0020</td>
<td>Executable</td>
</tr>
<tr>
<td>OB_READ</td>
<td>0x0010</td>
<td>Read permission</td>
</tr>
<tr>
<td>OB_USER</td>
<td>0x0008</td>
<td>User Storage</td>
</tr>
<tr>
<td>OB_WRITE</td>
<td>0x0004</td>
<td>Write permission</td>
</tr>
<tr>
<td>OB_HUGE</td>
<td>0x0002</td>
<td>Object is huge</td>
</tr>
<tr>
<td>OB_SHRINKABLE</td>
<td>0x0001</td>
<td>Object is Shrinkable</td>
</tr>
</tbody>
</table>
### ob_xflags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMOB_SLOCK_WAIT</td>
<td>0x01</td>
<td>Waiting on short term locks to clear</td>
</tr>
<tr>
<td>VMOB_LLOCK_WAIT</td>
<td>0x02</td>
<td>Waiting on long term locks to clear</td>
</tr>
<tr>
<td>VMOB_DISC_SEG</td>
<td>0x04</td>
<td>Object is part of a discardable seg</td>
</tr>
<tr>
<td>VMOB_HIGHMEM</td>
<td>0x08</td>
<td>Object was allocated via dh_vmalloc</td>
</tr>
</tbody>
</table>

---

### Virtual Page Structure

#### Pointers

pf_pvp points to the head of the VP array.

#### Locations

System Arena.

#### VM Owner

pgvp (0xffff)

#### Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Off</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avp_s</td>
<td>+0</td>
<td>c</td>
<td>S</td>
<td>active vp</td>
<td></td>
</tr>
<tr>
<td>vp_frame</td>
<td>+0</td>
<td>2.4</td>
<td>D</td>
<td>frame, swp or ldr block #</td>
<td></td>
</tr>
<tr>
<td>vp_flags</td>
<td>1.4</td>
<td></td>
<td></td>
<td>flags</td>
<td></td>
</tr>
<tr>
<td>vp_obpg</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>object relative page number</td>
<td></td>
</tr>
<tr>
<td>vp_hob</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>handle to object record</td>
<td></td>
</tr>
<tr>
<td>vp_refcount</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>virtual page reference count</td>
<td></td>
</tr>
<tr>
<td>vp_semowner</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>Slot number of semaphore owner</td>
<td></td>
</tr>
<tr>
<td>fvp_s</td>
<td>+0</td>
<td>a</td>
<td>S</td>
<td>Free vp</td>
<td></td>
</tr>
<tr>
<td>vp_flink</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>forward link</td>
<td></td>
</tr>
<tr>
<td>vp_blink</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>backward link</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>pad</td>
<td></td>
</tr>
</tbody>
</table>

#### vp_flag flag definitions:

<table>
<thead>
<tr>
<th>name</th>
<th>bit mask</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP_BUSY</td>
<td>0x001</td>
<td>page semaphore taken</td>
</tr>
<tr>
<td>VP_WANTED</td>
<td>0x002</td>
<td>page semaphore requested</td>
</tr>
</tbody>
</table>
Kernel Heap Header Structure

Pointers

_apkh points to the head of the VMKH array.
SAS_vm_heap_info also points to this array.

Locations

System Arena.

VM Owner

os2krnl (0xffaa)

Format

VMKH Kernel Heap Header.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset Length Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kh_fl</td>
<td>+0 4 D Allocation flags</td>
</tr>
<tr>
<td>kh_pkrh</td>
<td>+4 4 D Pointer to resident heap header</td>
</tr>
<tr>
<td>kh_pksh</td>
<td>+4 4 D Pointer to swappable heap header</td>
</tr>
</tbody>
</table>

kh_fl flag definitions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG_CONTIG</td>
<td>0x00000001</td>
<td>contiguous physical memory</td>
</tr>
<tr>
<td>PG_NOINCR</td>
<td>0x00000001</td>
<td>don't increment physical addr</td>
</tr>
<tr>
<td>PG_W</td>
<td>0x00000002</td>
<td>Writable - value from pte</td>
</tr>
<tr>
<td>PG_U</td>
<td>0x00000004</td>
<td>user mode accessible - from pte</td>
</tr>
<tr>
<td>PG_X</td>
<td>0x00000008</td>
<td>eXecutable</td>
</tr>
</tbody>
</table>

VP_CACHE 0x004 search page cache for pf
VP_PFIDLE 0x008 cross linked to idle pf
VP_PF 0x010 cross linked to pf
VP_DF 0x020 has swap file disk frame
VP_DIRTY 0x040 contents written to - from pte
VP_SHDIRTY 0x080 shadow dirty bit (for VDMs)
VP_SOW 0x100 change to swappable on write
VP_PRIVATIZED 0x200 vp privatized
VP_RESIDENT 0x400 cannot be moved - value from pte
VP_DISCARDABLE 0x800 1 = discardable, 0 = swappable
Heap Handles (hkh)

Kernel heap handles used to index the array of VMKH structures.

<table>
<thead>
<tr>
<th>Name</th>
<th>hkh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM_HKH_PUB_RESRW</td>
<td>1</td>
<td>Kernel resident RW heap handle</td>
</tr>
<tr>
<td>VM_HKH_PUB_RESRO</td>
<td>2</td>
<td>Public resident RO heap handle</td>
</tr>
<tr>
<td>VM_HKH_PUB_SWAPRW</td>
<td>3</td>
<td>Public swappable RW heap handle</td>
</tr>
<tr>
<td>VM_HKH_PUB_SWAPRO</td>
<td>4</td>
<td>Public swappable RO heap handle</td>
</tr>
<tr>
<td>VM_HKH_PUB_RES1MRW</td>
<td>5</td>
<td>Public resident RW 1M handle</td>
</tr>
<tr>
<td>VM_HKH_PUB_RES1MRO</td>
<td>6</td>
<td>Public resident RO 1M handle</td>
</tr>
</tbody>
</table>

Note:

It is possible for more than one handle to be served by the same heap. In particular, under the RETAIL kernel all heap handle are mapped to either a read/write resident or swappable heap.

Kernel Resident Heap Structures

Pointers

For resident heap entries, `kh_pkrh` of a VMKH entry points to a VMKRH.

An array of VMKRIHY structures are embedded in VMKRH at `krh_akrhy`.

An array of VMKRHS structures are embedded in VMKRH at `krh_akrhs`.

`krhf_pbNext` and `krhb_pbPrev` double link VMKRH structures from the dummy VMKRHF embedded in VMKRK at `krh_krhfDummy`.

Locations

System Arena.

VMKRH prefixes a resident heap.

Allocated blocks (VMKRHB and VMKRHBA) are sparsely allocated from the heap, with interstitial free blocks (VMKRHF).
VMKRHB prefixes the data portion of an allocated block.

VMKRHBA suffixes the data portion of an allocated block when the krh_attr bit is set in the associated VMKRHB.

**VM Owners**

- **vmkrhro (0xffeb)**
- **vmkrhrw (0xffec)**
- **kdbsym (0xff7c)**
- **krhrw1m (0xff43)**
- **krhrro1m (0xff44)**

**Format**

**VMKRH** Kernel Resident Heap Header.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>krh_cBlocks</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Count of heap blocks</td>
</tr>
<tr>
<td>krh_cFreeBlocks</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Count of free heap blocks</td>
</tr>
<tr>
<td>krh_pbFirst</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>First Heap Block</td>
</tr>
<tr>
<td>krh_pbLast</td>
<td>+C</td>
<td>4</td>
<td>D</td>
<td>Last Heap Block</td>
</tr>
<tr>
<td>krh_pbEndRes</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Upper bound of reserved virt mem</td>
</tr>
<tr>
<td>krh_pbEndCom</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Upper bound of committed virt mem</td>
</tr>
<tr>
<td>krh_lpgBase</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Start lin page of heap object</td>
</tr>
<tr>
<td>krh_cMods</td>
<td>+1C</td>
<td>4</td>
<td>D</td>
<td>Count of heap modifications</td>
</tr>
<tr>
<td>krh_fl</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Resident heap flags</td>
</tr>
<tr>
<td>krh_krhfDummy</td>
<td>+24</td>
<td>C</td>
<td>D</td>
<td>Dummy free block</td>
</tr>
<tr>
<td>krh_krhfLast</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Last Freelist Section Pointer</td>
</tr>
<tr>
<td>krh_akrhy</td>
<td>+34</td>
<td>30</td>
<td>S</td>
<td>Array of 8 yield structures</td>
</tr>
<tr>
<td>krh_akrhs</td>
<td>+64</td>
<td>50</td>
<td>S</td>
<td>Array of 5 free list sections</td>
</tr>
<tr>
<td>krh_hob</td>
<td>+B4</td>
<td>2</td>
<td>W</td>
<td>Heap object handle</td>
</tr>
<tr>
<td>krh_ksem</td>
<td>+B6</td>
<td>C(10)</td>
<td>S</td>
<td>KSEM for resident heap</td>
</tr>
</tbody>
</table>

**VMKRHY** Kernel Resident Heap Yield List Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>krhy_pb</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Block pointer</td>
</tr>
<tr>
<td>krhy_cyield</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Yield count</td>
</tr>
</tbody>
</table>

**VMKRHS** Kernel Resident Heap Free List Anchor Structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>krhs_cbMax</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Maximum block size</td>
</tr>
<tr>
<td>krhs_krhfHead</td>
<td>+4</td>
<td>C</td>
<td>S</td>
<td>Dummy free block used to locate head of list</td>
</tr>
</tbody>
</table>
VMKRHF Kernel Resident Heap Free Block.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>krhf_struct0</td>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>Resident Heap Block Header</td>
</tr>
<tr>
<td>krhf_pbNext</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Forward freelist link</td>
</tr>
<tr>
<td>krhf_pbPrev</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Backward freelist link</td>
</tr>
</tbody>
</table>

VMKRHB Kernel Resident Heap Block Header.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>Regular Block Header</td>
<td></td>
</tr>
<tr>
<td>krhb_hobowner</td>
<td>+0</td>
<td>2</td>
<td>D</td>
<td>Owner</td>
</tr>
<tr>
<td>krhb_pfree</td>
<td>0.1</td>
<td></td>
<td>Preceding block is free</td>
<td></td>
</tr>
<tr>
<td>krhb_usSize</td>
<td>1.5</td>
<td></td>
<td>Size of block in dwords</td>
<td></td>
</tr>
<tr>
<td>krhb_yield</td>
<td>0.1</td>
<td></td>
<td>Thread-yielded-here flag</td>
<td></td>
</tr>
<tr>
<td>krhb_attr</td>
<td>0.1</td>
<td></td>
<td>Attributed block flag (=0)</td>
<td></td>
</tr>
<tr>
<td>+0</td>
<td>4</td>
<td>S</td>
<td>Attributed Block Header</td>
<td></td>
</tr>
<tr>
<td>+0</td>
<td>0.1</td>
<td>D</td>
<td>Attributed block is free</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td></td>
<td>Block preceding attributed block is free</td>
<td></td>
</tr>
<tr>
<td>krhb_ulSize</td>
<td>3.4</td>
<td></td>
<td>Size of block in dwords</td>
<td></td>
</tr>
<tr>
<td>krhb_yield</td>
<td>0.1</td>
<td></td>
<td>Thread-yielded-here flag</td>
<td></td>
</tr>
<tr>
<td>krhb_attr</td>
<td>0.1</td>
<td></td>
<td>Attributed block flag (=1)</td>
<td></td>
</tr>
</tbody>
</table>

VMKRHBA Kernel Resident Heap Block Header Attributes.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>khba_sel</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Selector mapping heap block data</td>
</tr>
<tr>
<td>khba_hobOwner</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Heap block owner</td>
</tr>
<tr>
<td>khba_hobMTE</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Heap block MTE</td>
</tr>
<tr>
<td>khba_pad</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Pad to DWORD multiple</td>
</tr>
</tbody>
</table>

krh_fl flag definitions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG_CONTIG</td>
<td>0x00000001</td>
<td>contiguous physical memory</td>
</tr>
<tr>
<td>PG_NOINCR</td>
<td>0x00000001</td>
<td>don't increment physical addr</td>
</tr>
<tr>
<td>PG_W</td>
<td>0x00000002</td>
<td>Writable - value from pte</td>
</tr>
<tr>
<td>PG_U</td>
<td>0x00000004</td>
<td>user mode accessible - from pte</td>
</tr>
<tr>
<td>PG_X</td>
<td>0x00000008</td>
<td>executable</td>
</tr>
<tr>
<td>PG_R</td>
<td>0x00000010</td>
<td>readable</td>
</tr>
</tbody>
</table>
Kernel Swappable Heap Structures

Pointers

For swappable heap entries, `kh_pksh` of a VMKH entry points to a VMKSH.

An array of VMKRHY structures are embedded in VMKRH at `krh_akrhy`.

An array of VMKRHS structures are embedded in VMKRH at `krh_akrhs`.

`ksh_hdrEntry` in VMKSH points to the first chained VMKSHD.

`kshd_pbNext` in VMKSHD points to subsequent VMKSHD structures. VMKRHF structures from the dummy VMKRHF embedded in VMKRK at `krh_krhfDummy`.

Locations

System Arena.

VMKSHD structure are allocated from the Kernel Resident Heap.

VMKSH prefixes a swappable heap.

Allocated blocks (VMKSHB) are sparsely allocated from the heap.

VMKSHB prefixes the data portion of an allocated block.

VM Owners

VMKSH and VMKSHB:

- `vmkshro` (0xffee)
- `vmkshrw` (0xffef).

VMKSHD:

- `vmshd` (0xffed).

Format

VMKSH Kernel Swappable Heap Header (RETAIL kernel)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ksh_hdrEntry</td>
<td>+0</td>
<td>12</td>
<td>S</td>
<td>Dummy head descriptor of free chain</td>
</tr>
<tr>
<td></td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
VMKSH Kernel Swappable Heap Header (ALLSTRICT kernel)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ksh_hdrEntry</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Head of descriptor chain</td>
</tr>
<tr>
<td>ksh_pHint</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Descriptor pointer to last block touched</td>
</tr>
<tr>
<td>ksh_pHPrev</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Descriptor pointer to PrevBlk of last touched</td>
</tr>
<tr>
<td>ksh_HintSize</td>
<td>+C</td>
<td>4</td>
<td>D</td>
<td>Size of last block touched</td>
</tr>
<tr>
<td>ksh_pbEndRes</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Upper bound of reserved virt mem</td>
</tr>
<tr>
<td>ksh_pbEndCom</td>
<td>+1C</td>
<td>4</td>
<td>D</td>
<td>Upper bound of committed virt mem</td>
</tr>
<tr>
<td>ksh_f1</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Swappable heap allocation flags</td>
</tr>
<tr>
<td>ksh_hob</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Heap object handle</td>
</tr>
<tr>
<td>ksh_pbStart</td>
<td>+26</td>
<td>4</td>
<td>D</td>
<td>Lower bound of reserved virt mem</td>
</tr>
<tr>
<td>ksh_ksem</td>
<td>+2A</td>
<td>C</td>
<td>S</td>
<td>KSEM for swappable heap</td>
</tr>
</tbody>
</table>

VMKSHD Kernel Swappable Heap Descriptor Record for Free Blocks.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kshb_size</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Size of block in dwords</td>
</tr>
<tr>
<td>kshd_pNext</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Next free block</td>
</tr>
<tr>
<td>kshd_pb</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Address of block header</td>
</tr>
</tbody>
</table>

VMKSHB Kernel Swappable Heap Block Header for Allocated blocks.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
+0  1  D  Signature 0x52
kshb_size   3  size of block
kshb_hobowner  +4  2  W  Owner
kshb_sel  +6  2  W  Selector

**ksh_fl flag definitions.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG_CONTIG</td>
<td>0x00000001</td>
<td>contiguous physical memory</td>
</tr>
<tr>
<td>PG_NOINCR</td>
<td>0x00000001</td>
<td>don't increment physical addr</td>
</tr>
<tr>
<td>PG_W</td>
<td>0x00000002</td>
<td>Writable - value from pte</td>
</tr>
<tr>
<td>PG_U</td>
<td>0x00000004</td>
<td>user mode accessible - from pte</td>
</tr>
<tr>
<td>PG_X</td>
<td>0x00000008</td>
<td>eXecutable</td>
</tr>
<tr>
<td>PG_R</td>
<td>0x00000010</td>
<td>Readable</td>
</tr>
<tr>
<td>PG_1M</td>
<td>0x00000020</td>
<td>must reside below 1 meg physical</td>
</tr>
<tr>
<td>PG_GUARD</td>
<td>0x00000040</td>
<td>guard page - from pte</td>
</tr>
<tr>
<td>PG_16M</td>
<td>0x00000040</td>
<td>must reside below 16 meg physical</td>
</tr>
<tr>
<td>PG_ZEROFILL</td>
<td>0x00000080</td>
<td>zero initialize pages</td>
</tr>
<tr>
<td>PG_SWAPONWRITE</td>
<td>0x00000100</td>
<td>value from vp</td>
</tr>
<tr>
<td>PG_UVIRT</td>
<td>0x00000200</td>
<td>value from pte</td>
</tr>
<tr>
<td>PG_RESIDENT</td>
<td>0x00000400</td>
<td>value from pte</td>
</tr>
<tr>
<td>PG_DISCARDABLE</td>
<td>0x00000800</td>
<td>value from vp</td>
</tr>
</tbody>
</table>

--------------------------------------------

**Scheduler Thread and Process Control Block Reference**

The following control blocks are described in this section:

- Thread Control Block (TCB)
- Thread Swappable Data (TSD)
- Per Task Data Area (PTDA)
- The local exception handler Long-Jump Buffer (ljmp)
- Local Information Segment (LISEG)
- Global Information Segment (GISEG)
- Process Information Block (PIB)
- Thread Information Block (TIB)
- System Stack Frames and Client Register Information
- Exit List Data Structure (EXENT)
Exception Handler Structures

An overview of the Scheduler Control Blocks follows:

---

Scheduler and Task Management Control Block Diagrams

The following diagrams illustrate the relationships between various Scheduler and Task Management control blocks:

- Process Management
- Thread Management
- Scheduler Finite State Machine
- Thread Tree for a Process
- Process Trees, Subtrees and Zombies
- Orphaned and Adopted Processes
- Exception Management - Overview
- Exception Handler Stack Frames

---

Process Management
Thread Management
Scheduler Finite State Machine

Thread Tree for a Process
Thread Tree for a Process

Thread 3 is in critical section

Thread 2 and Thread 1 are waiting for Thread 4 to die
The Process Tree, Subtrees and Zombies

Pid 1 is Detached (no parent)

Other Detached Processes are Siblings of Pid 1

Pids 1 - 7 are active

Pids 8, 11 are dead (zombies)

Pid 4 may DoWaitChild on Pids 8 - 10

Pid 5 may DoWaitChild on Pid 11
Orphaned and Adopted Processes
Orphaned and Adopted Processes

PTDA

CSID 2

CSID 5

CSID 6

Pid 2 dies

CSID 2
OS/2 Exception Management - Overview
Exception Handling - Overview

Interrupt Descriptor Table

Specific 1st Level Trap Handlers Are Entered
(trap0, trap1, ...)

Fault Handled?

Operating System

TrapCommonFaultEntry

Trace etc

VMX Fault

Process Fault

Kernel Fault

Enter Debugger

for VSF/VTM

Call Delay/HardErr

Asynchronous Notification

_xcpExceptionCallback

Kernel Mode

_user Mode

Jos32ExceptionDispatcher

Exception Handler

Jos32ExceptionCallback

Terminate

Continue

Call DD SFE Exits

Re-enter Panic (TPR)
Exception Handler Stack Frames
Exception Handler Stack Frames

low address

exception handler ESP

Exception Reg Rec

Exception Reg Rec

high address

This exception frame is repeated for nested No exceptions
Thread Control Block OS/2 Warp V4.0

For **TCB** formats for other versions of OS/2 see:

- **TCB** for OS/2 Warp V3.0
- **TCB** for OS/2 Warp V3.0 with Fix pack 9
- **TCB** for OS/2 Warp V3.0 with Fix pack 11 or later
- **TCB** for OS/2 V2.11 with Fix pack 90 or later
- **TCB** for OS/2 V2.11

**Pointers**

- _papTCBSlots_ points to the thread slot table of TCB pointers.

  Multiple chain pointers between, TSD, TCB and PTDA.

  **CurrTCB** points to the current TCB.

**Locations**

System Arena.

**VM Owner**

tcb (0xffcc)

**Format**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBOrdinal</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Ordinal number of thread in PTDA</td>
</tr>
<tr>
<td>TCBNumber</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread slot number</td>
</tr>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBptib</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>forward link to next (active) TCB</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBpStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBpStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBpKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBpKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBpKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBpKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBpKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
<tr>
<td>TCBpKStackPTDA2</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>KStack page 2 of PTDA</td>
</tr>
<tr>
<td>TCBCurrrTCB</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TCB</td>
</tr>
<tr>
<td>TCBCurrrTSD</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TSD</td>
</tr>
<tr>
<td>TCBBiasTCB</td>
<td>+6c</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TCB conversion value</td>
</tr>
<tr>
<td>TCBBiasTSD</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TSD conversion value</td>
</tr>
<tr>
<td>TCBpDIHRetAddr</td>
<td>+74</td>
<td>4</td>
<td>D</td>
<td>82818 Pointer to DHRouter return address</td>
</tr>
<tr>
<td>TCBTLMA</td>
<td>+78</td>
<td>80</td>
<td>D</td>
<td>Thread local memory area</td>
</tr>
<tr>
<td>TCBDMAAdd</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>Position of first sector accessed within file</td>
</tr>
<tr>
<td>TCBThisSFT</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>pointer to SFT we're working with</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Number of valid (previously written) sectors</td>
</tr>
<tr>
<td>TCBpRTCB</td>
<td>+108</td>
<td>4</td>
<td>D</td>
<td>Redirector TCB (Used by LANMAN)</td>
</tr>
<tr>
<td>TCBProc_ID</td>
<td>+10c</td>
<td>2</td>
<td>W</td>
<td>process ID for file sharing checks</td>
</tr>
<tr>
<td>TCBUser_ID</td>
<td>+10e</td>
<td>2</td>
<td>W</td>
<td>user ID for file sharing checks</td>
</tr>
<tr>
<td>TCfSharing</td>
<td>+110</td>
<td>1</td>
<td>B</td>
<td>non-zero ==&gt; no redirection</td>
</tr>
<tr>
<td>TCBSrvAttrib</td>
<td>+111</td>
<td>1</td>
<td>B</td>
<td>see SetAttrib/file.asm</td>
</tr>
<tr>
<td>TCBJfnFlag</td>
<td>+112</td>
<td>1</td>
<td>B</td>
<td>JFN flag bits for current fil handle</td>
</tr>
<tr>
<td>TCBAAllowed</td>
<td>+113</td>
<td>1</td>
<td>B</td>
<td>Allowed I 24 answers (see allowed_)</td>
</tr>
<tr>
<td>TCBopCookie</td>
<td>+114</td>
<td>4</td>
<td>D</td>
<td>server's per file cookie</td>
</tr>
<tr>
<td>TCBopFlags</td>
<td>+118</td>
<td>2</td>
<td>W</td>
<td>whether server wants oplock, etc.</td>
</tr>
<tr>
<td>TCBCurBuf</td>
<td>+11a</td>
<td>4</td>
<td>D</td>
<td>currently assigned buffer</td>
</tr>
<tr>
<td>TCBThisVPB</td>
<td>+11e</td>
<td>2</td>
<td>W</td>
<td>handle of current VPB</td>
</tr>
<tr>
<td>TCBNextAdd</td>
<td>+120</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytSecPos</td>
<td>+122</td>
<td>2</td>
<td>W</td>
<td>position of first byte within sector</td>
</tr>
<tr>
<td>TCBClusNum</td>
<td>+124</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBLastPos</td>
<td>+126</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytCnt1</td>
<td>+128</td>
<td>2</td>
<td>W</td>
<td>Number of bytes in 1st sector</td>
</tr>
<tr>
<td>TCBBytCnt2</td>
<td>+12a</td>
<td>2</td>
<td>W</td>
<td># of bytes in last sector</td>
</tr>
<tr>
<td>TCBSecCnt</td>
<td>+12c</td>
<td>2</td>
<td>W</td>
<td>number of whole sectors</td>
</tr>
<tr>
<td>TCBSecClusPos</td>
<td>+12e</td>
<td>1</td>
<td>B</td>
<td>position of first sector within cluster</td>
</tr>
<tr>
<td>TCBNoSetDir</td>
<td>+12f</td>
<td>1</td>
<td>B</td>
<td>If TRUE, do not set directory</td>
</tr>
<tr>
<td>TCBJoins</td>
<td>+130</td>
<td>1</td>
<td>B</td>
<td>number of joins</td>
</tr>
<tr>
<td>Symbol</td>
<td>Offset</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>TCBPad</td>
<td>+131</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCBDevFCB</td>
<td>+132</td>
<td>B</td>
<td>Uses Name1, Name2, combined</td>
<td></td>
</tr>
<tr>
<td>TCB_direntry</td>
<td>+133</td>
<td>20</td>
<td>Directory entry</td>
<td></td>
</tr>
<tr>
<td>dir_name</td>
<td>+133</td>
<td>b</td>
<td>File name</td>
<td></td>
</tr>
<tr>
<td>dir_attr</td>
<td>+13e</td>
<td>1</td>
<td>Attribute bits</td>
<td></td>
</tr>
<tr>
<td>dir_pad</td>
<td>+13f</td>
<td>8</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>dir_EAhandle</td>
<td>+147</td>
<td>2</td>
<td>First cluster of extended attribute</td>
<td></td>
</tr>
<tr>
<td>dir_time</td>
<td>+149</td>
<td>2</td>
<td>Time of last write</td>
<td></td>
</tr>
<tr>
<td>dir_date</td>
<td>+14b</td>
<td>2</td>
<td>Date of last write</td>
<td></td>
</tr>
<tr>
<td>dir_firstfile</td>
<td>+14d</td>
<td>2</td>
<td>First allocation unit of file</td>
<td></td>
</tr>
<tr>
<td>dir_size_l</td>
<td>+14f</td>
<td>2</td>
<td>Low 16 bits of file size</td>
<td></td>
</tr>
<tr>
<td>dir_size_h</td>
<td>+151</td>
<td>2</td>
<td>High 16 bits of file size</td>
<td></td>
</tr>
<tr>
<td>TCBName1</td>
<td>+153</td>
<td>c</td>
<td>File name buffer <em>REDIR</em></td>
<td></td>
</tr>
<tr>
<td>TCBName2</td>
<td>+15f</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCBDESTSTART</td>
<td>+16c</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCBDirPad</td>
<td>+16d</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCBBufHE</td>
<td>+173</td>
<td>B</td>
<td>How to handle a HardError</td>
<td></td>
</tr>
<tr>
<td>TCBBactBufHE</td>
<td>+174</td>
<td>B</td>
<td>action response from user on HardErr</td>
<td></td>
</tr>
<tr>
<td>TCBfIOLock</td>
<td>+175</td>
<td>B</td>
<td>NZ if TCBLockHndl1 is valid</td>
<td></td>
</tr>
<tr>
<td>TCBLockHndl1</td>
<td>+176</td>
<td>C</td>
<td>Lock handle of user mem</td>
<td></td>
</tr>
<tr>
<td>TCBThisCDS</td>
<td>+182</td>
<td>4</td>
<td>Address of current CDS</td>
<td></td>
</tr>
<tr>
<td>TCBThisFSC</td>
<td>+186</td>
<td>4</td>
<td>address of current FSC</td>
<td></td>
</tr>
<tr>
<td>TCBpTempCDS</td>
<td>+18a</td>
<td>4</td>
<td>Address of dummycds</td>
<td></td>
</tr>
<tr>
<td>TCBpOpenBuf</td>
<td>+18e</td>
<td>2</td>
<td>Address of current OpenBuf</td>
<td></td>
</tr>
<tr>
<td>TCBpSearchBuf</td>
<td>+190</td>
<td>2</td>
<td>Address of SearchBuf</td>
<td></td>
</tr>
<tr>
<td>TCBFailErr</td>
<td>+192</td>
<td>2</td>
<td>NZ if user did FAIL on I 24</td>
<td></td>
</tr>
<tr>
<td>TCBShareRetriesLeft</td>
<td>+194</td>
<td>2</td>
<td>number of share/lock viol retries</td>
<td></td>
</tr>
<tr>
<td>TCBRetryCount</td>
<td>+196</td>
<td>2</td>
<td>num of share/lock retries to do</td>
<td></td>
</tr>
<tr>
<td>TCBRetryLoop</td>
<td>+198</td>
<td>2</td>
<td>num of share/lock retry delay loops</td>
<td></td>
</tr>
<tr>
<td>TCB_pScrchBuf</td>
<td>+19a</td>
<td>2</td>
<td>internal search buffer</td>
<td></td>
</tr>
<tr>
<td>TCB_pOpenBuf</td>
<td>+19c</td>
<td>2</td>
<td>Pointer to a scratch buffer on stack</td>
<td></td>
</tr>
<tr>
<td>TCBAttrib</td>
<td>+19e</td>
<td>2</td>
<td>storage for file attributes <em>REDIR</em></td>
<td></td>
</tr>
<tr>
<td>TCBExtFCB</td>
<td>+1a0</td>
<td>B</td>
<td>Extended FCB</td>
<td></td>
</tr>
<tr>
<td>TCBpad2</td>
<td>+1a1</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCBWFP_Start</td>
<td>+1a2</td>
<td>W</td>
<td>TASKAREA offset for working string <em>REDIR</em></td>
<td></td>
</tr>
<tr>
<td>TCBRen_WFP</td>
<td>+1a4</td>
<td>2</td>
<td>WFB pointer for rename destination <em>REDIR</em></td>
<td></td>
</tr>
<tr>
<td>TCBWFP_Path_End</td>
<td>+1a6</td>
<td>2</td>
<td>End of Path component of string</td>
<td></td>
</tr>
<tr>
<td>TCBCurr_Dir_End</td>
<td>+1a8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
TCBDTAddr +1aa 4 D User's I/O transfer address *REDIR*
TCBVolID +1ae 1 B !0 if vol ID found in dir search
TCBSpaceFlag +1af 1 B Embedded spaces allowed in FCB
TCBCreating +1b0 1 B
TCBDelAll +1b1 1 B
TCBFoundDel +1b2 1 B
TCBFound_dev +1b3 1 B true => search found a device 3.10
TCBFsplice +1b4 1 B true => do a splice in transpath 3.10
TCBClusFac +1b5 1 B sectors/cluster used in dir search
TCBcMeta +1b6 1 B components found 3.10
TCBPathNameType +1b7 1 B
TCBDevPt +1b8 4 D Address of device found by DevName *REDIR*
TCBDirSec +1bc 4 D Variables used in directory searching
TCBDirStart +1c0 2 W Variables used in directory searching
TCBNextClusNum +1c2 2 W Variables used in directory searching
TCBEntFree +1c4 2 W Variables used in directory searching
TCBEntLast +1c6 2 W Variables used in directory searching
TCBLastEnt +1c8 2 W Variables used in directory searching
TCBSattrib +1ca 2 W Storage for search attrs *REDIR* 3.10
TCB_SemInfo +1cc 4 D 16bit addr of the ramsem blocked upon
TCB_SemDebugAddr +1d0 4 D debugger display address for ksems
TCB_NPX_Buffer +1d4 4 D
TCBP�件TCBWaitNext +1d8 4 D Next waiting TCB
TCBP�件TCBWaitList +1dc 4 D Threads waiting for me to die
TCBSQstate +1e0 1 B Scheduler queue location (actual)
TCBState +1e1 1 B Current scheduler state (desired)
TCBWakeFlags +1e2 1 B TKSleep/TKWakeUp Flags
TCBWindowBoost +1e3 1 B Window Boost count
TCBPriorityClass +1e4 1 B Priority Class (user)
TCBPriLevel +1e5 1 B Priority Level (user)
TCBPriClassMod +1e6 1 B Priority Class modifier bits
TCBSchFlags +1e7 1 B Misc. Scheduler flags
TCBPriority +1e8 2 W Calculated Priority
TCBPriorityMin +1ea 2 W Minimum Scheduling priority
TCBcBoostLock +1ec 4 D Kernel Boost Lock nesting count.
TCBP�件TCBPriNextQ +1fo 4 D Next priority queue in chain
TCBP�件TCBPriPrevQ +1f4 4 D Previous priority queue in chain
TCBP�件TCBPriHigher +1f8 4 D Higher priority thread
TCBP�件TCBPriLower +1fc 4 D Lower priority thread
TCBpTCBPriNext      +200   4      D    Next same-priority thread
TCBpTCBPriPrev      +204   4      D    Prev same-priority thread
TCBpTCBWakeup       +208   4      D    TKQueryWakeup TCB list
TCBSleepID          +20c   4      D    Sleep ID this TCB is sleeping on
TCBtoe              +210   14     S    Timeout/Starvation Timeout element
TCBCheckedSig       +224   1      B    Used by the loader
TCBsSwapping        +225   1      B    status of swapping
TCBVigIONest        +226   1      B    nesting level of FSH_DoVolIO
TCBReqPktFlg        +227   1      B    Flag to indicate if request pkt in use
TCBReqPkt           +228   4      D    I/O request packet for thread
TCBSysTime          +22c   4      D    time spent in system code
TCBUserTime         +230   4      D    time spent in user code
TCB_pPVDBThd        +234   4      D    Ptr to Perfview Data Block for this
                            thread (pvdb_thd_s).
TCB_flDbg           +238   4      D
TCBCpl2_ESP          +23c   4      D    Saved TSS CPL2 stack pointer.
TCBCpl2_SS          +240   2      W    Saved TSS CPL2 stack segment.
TCBNewFlags         +242   1      B    Value copied from ptda_NewFiles
TCBEntryActions     +243   1      B    Kernel entry force flags
TCBSig_pend         +244   2      W    bit vector of pending signals
TCBSig_holding      +246   2      W    bit vector of postponed signals
TCBSig_cur          +248   2      W    bit vec of signals being processed
TCBXcptRepRec       +24a   4      D    report record of active exception
TCBSig_termtid      +24e   2      W    tid of terminator -75797
TCB_Secbits         +250   1      B    Security bits 54735
TCB_spbytes         +251   1      B    To keep size 4*N 54735
TCB_ulSRIndex       +252   4      D    Last semaphore cleared in MUX 72485
TCBMiscFlags        +256   1      B    Used for hard error processing
TCBModeFlags        +257   2      W    Mode flags for OPEN - for WhatVolume
TCBSpareFlags       +259   1      B    Spare flags
TCBLibiFlags        +25a   1      B    84537
TCBFiller           +25b   1      B    To keep size 4*N
TCB_ProcNameBuf     +25c   4      D    Pointer to procedure name
TCB_ObjNameBuf      +260   4      D    Pointer to object name buffer
TCB_TmpNameBuf      +264   4      D    aka TCB_TgtModNameBuf
TCB_SrcModNameBuf   +268   4      D    Used by loader
TCB_FaultBuf        +26c   4      D
TCB_ObjNameBufL     +270   2      W    Length of object name buffer
TCB_TmpNameBufL     +272   2      W
TCB_SrcModNameBufL  +274   2      W
TCB_FaultBufL  +276  2  W
TCBSecchild  +278  4  D  Child Security data 54735

**TCB Forced Actions** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK_FF_BUF</td>
<td>0x00000001</td>
<td>Buffer must be released</td>
</tr>
<tr>
<td>TK_FF_EXIT</td>
<td>0x00000002</td>
<td>Call TKExit (old FF_DES)</td>
</tr>
<tr>
<td>TK_FF_CRITSEC</td>
<td>0x00000004</td>
<td>Enter Per-task critical section</td>
</tr>
<tr>
<td>TK_FF_ICE</td>
<td>0x00000008</td>
<td>Freeze thread</td>
</tr>
<tr>
<td>TK_FF_NPX</td>
<td>0x00000010</td>
<td>NPX Error</td>
</tr>
<tr>
<td>TK_FF_TIB</td>
<td>0x00000020</td>
<td>Update the TIB</td>
</tr>
<tr>
<td>TK_FF_TRC</td>
<td>0x00000040</td>
<td>Enter Debug</td>
</tr>
<tr>
<td>TK_FF_SIG</td>
<td>0x00000080</td>
<td>Signal pending</td>
</tr>
<tr>
<td>TK_FF_CTXH</td>
<td>0x00000100</td>
<td>Pending local context hooks</td>
</tr>
<tr>
<td>TK_FF_STIH</td>
<td>0x00000200</td>
<td>Execute STI hooks</td>
</tr>
<tr>
<td>TK_FF_VDMBP</td>
<td>0x00000400</td>
<td>Execute VDM BP hooks</td>
</tr>
<tr>
<td>TK_FF_RTRY</td>
<td>0x00000800</td>
<td>Retry V86 system call</td>
</tr>
<tr>
<td>TK_FF_PIB</td>
<td>0x00001000</td>
<td>Update the PIB</td>
</tr>
<tr>
<td>TK_FF_SCH</td>
<td>0x00002000</td>
<td>Do Scheuler Processing</td>
</tr>
<tr>
<td>TK_FF_TFBIT</td>
<td>0x00004000</td>
<td>Validate user eflags TF bit</td>
</tr>
<tr>
<td>TK_FF_TIBPRI</td>
<td>0x00008000</td>
<td>Update only the priority fields in TIB</td>
</tr>
</tbody>
</table>

59463

**TCB Entry Actions** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK_EF_PFCLI</td>
<td>1</td>
<td>Page fault inside CLI</td>
</tr>
<tr>
<td>TK_EF_TRC</td>
<td>2</td>
<td>DosDebug action pending</td>
</tr>
</tbody>
</table>

**TCB Wake Flags** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK_WF_INTERRUPTED</td>
<td>0x01</td>
<td>Sleep was interrupted</td>
</tr>
<tr>
<td>TK_WF_TIMEEXP</td>
<td>0x02</td>
<td>Timeout expired</td>
</tr>
<tr>
<td>TK_WF_INTENDING</td>
<td>0x04</td>
<td>Interrupt pending</td>
</tr>
<tr>
<td>TK_WF_SINGLEWAKEUP</td>
<td>0x08</td>
<td>Thread wants single wakeup</td>
</tr>
<tr>
<td>TK_WF_INTERRUPTIBLE</td>
<td>0x10</td>
<td>Thread blocked interruptibly</td>
</tr>
<tr>
<td>TK_WF_TIMEOUT</td>
<td>0x20</td>
<td>Thread blocked with timeout</td>
</tr>
<tr>
<td>TK_WF_SLEEPING</td>
<td>0x40</td>
<td>In TKSleep()</td>
</tr>
</tbody>
</table>
TCBState and TCBQState definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE_VOID</td>
<td>0</td>
<td>Uninitialized</td>
</tr>
<tr>
<td>STATE_READY</td>
<td>1</td>
<td>Ready to run</td>
</tr>
<tr>
<td>STATE_BLOCKED</td>
<td>2</td>
<td>Blocked on an ID</td>
</tr>
<tr>
<td>STATE_SUSPENDED</td>
<td>3</td>
<td>Suspended (DosSuspendThread)</td>
</tr>
<tr>
<td>STATE_CRITSEC</td>
<td>4</td>
<td>Blocked by another CritSec thread</td>
</tr>
<tr>
<td>STATE_RUNNING</td>
<td>5</td>
<td>Thread currently running</td>
</tr>
<tr>
<td>STATE_READYBOOST</td>
<td>6</td>
<td>Ready, but apply an IO boost</td>
</tr>
<tr>
<td>STATE_TSD</td>
<td>7</td>
<td>Thread waiting for TSD</td>
</tr>
<tr>
<td>STATE_DELAYED</td>
<td>8</td>
<td>Delayed TKWakeup (Almost Ready)</td>
</tr>
<tr>
<td>STATE_FROZEN</td>
<td>9</td>
<td>Frozen Thread (FF_ICE)</td>
</tr>
<tr>
<td>STATE_GETSTACK</td>
<td>10</td>
<td>Incomming TSD</td>
</tr>
<tr>
<td>STATE_BADSTACK</td>
<td>11</td>
<td>TSD failed to swap in</td>
</tr>
</tbody>
</table>

TCBPriClassMod definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSMOD_KEYBOARD</td>
<td>0x04</td>
<td>Keyboard boost</td>
</tr>
<tr>
<td>CLASSMOD_STARVED</td>
<td>0x08</td>
<td>Starvation boost</td>
</tr>
<tr>
<td>CLASSMOD_DEVICE</td>
<td>0x10</td>
<td>Device I/O Done Boost</td>
</tr>
<tr>
<td>CLASSMOD_FOREGROUND</td>
<td>0x20</td>
<td>Foreground boost</td>
</tr>
<tr>
<td>CLASSMOD_WINDOW</td>
<td>0x40</td>
<td>Window Boost</td>
</tr>
<tr>
<td>CLASSMOD_VDM_INTERRUPT</td>
<td>0x80</td>
<td>VDM simulated interrupt boost</td>
</tr>
</tbody>
</table>

TCBPriClass definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS_NOCHANGE</td>
<td>0x00</td>
<td>No priority class change</td>
</tr>
<tr>
<td>CLASS_IDLE_TIME</td>
<td>0x01</td>
<td>Idle-Time class</td>
</tr>
<tr>
<td>CLASS_REGULAR</td>
<td>0x02</td>
<td>Regular class</td>
</tr>
<tr>
<td>CLASS_TIME_CRITICAL</td>
<td>0x03</td>
<td>Time-Critical class</td>
</tr>
<tr>
<td>CLASS_SERVER</td>
<td>0x04</td>
<td>Client-Server Server class</td>
</tr>
</tbody>
</table>

TCBSchFlg flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCH_PROTECTED_PRI</td>
<td>0x0001</td>
<td>Only Intra-process SetPri allowed</td>
</tr>
<tr>
<td>SCH_WINDOWBOOST_LOCK</td>
<td>0x0002</td>
<td>Lock out windoboom changes</td>
</tr>
</tbody>
</table>
Thread Control Block for OS/2 Warp V3.0
<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBOrdinal</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Ordinal number of thread in PTDA</td>
</tr>
<tr>
<td>TCBNumber</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread slot number</td>
</tr>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBpStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBpStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBpteKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBpteKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA2</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>KStack page 2 of PTDA</td>
</tr>
<tr>
<td>TCBCurrTCB</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TCB</td>
</tr>
<tr>
<td>TCBcurrTSD</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TSD</td>
</tr>
<tr>
<td>TCBBiasTCB</td>
<td>+6c</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TCB conversion value</td>
</tr>
<tr>
<td>TCBBiasTSD</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TSD conversion value</td>
</tr>
<tr>
<td>TCBTLMA</td>
<td>+74</td>
<td>80</td>
<td>D</td>
<td>Thread local memory area</td>
</tr>
<tr>
<td>TCBDMAAdd</td>
<td>+f4</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>Position of first sector accessed within file</td>
</tr>
<tr>
<td>TCBThisSFT</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>pointer to SFT we're working with</td>
</tr>
<tr>
<td>TCBValSec</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>Number of valid (previously written) sectors</td>
</tr>
<tr>
<td>TCBpRTCB</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Redirector TCB (Used by LANMAN)</td>
</tr>
<tr>
<td>TCBProc_ID</td>
<td>+108</td>
<td>2</td>
<td>W</td>
<td>process ID for file sharing checks</td>
</tr>
<tr>
<td>TCBUser_ID</td>
<td>+10a</td>
<td>2</td>
<td>W</td>
<td>user ID for file sharing checks</td>
</tr>
<tr>
<td>TCBfSharing</td>
<td>+10c</td>
<td>1</td>
<td>B</td>
<td>non-zero ==&gt; no redirection</td>
</tr>
<tr>
<td>TCBsrvAttrib</td>
<td>+10d</td>
<td>1</td>
<td>B</td>
<td>see SetAttrib/file.asm</td>
</tr>
</tbody>
</table>
TCBJFnFlag          +10e  1  B  JFN flag bits for current fil handle
TCBAllowed          +10f  1  B  Allowed I 24 answers (see allowed_)
TCBOpCookie         +110  4  D  server's per file cookie
TCBOpFlags          +114  2  W  whether server wants oplock, etc.
TCBCurBuf           +116  4  D  currently assigned buffer
TCBThishVPB         +11a  2  W  handle of current VPB
TCBNextAdd          +11c  2  W
TCBBytSecPos        +11e  2  W  position of first byte within sector
TCBClusNum          +120  2  W
TCBLastPos          +122  2  W
TCBBytCnt1          +124  2  W  Number of bytes in 1st sector
TCBBytCnt2          +126  2  W  # of bytes in last sector
TCBSecCnt           +128  2  W  number of whole sectors
TCBSec1lusPos       +12a  1  B  posit of first sector within cluster
TCBufHE             +12b  1  B  How to handle a HardError
TCBactBufHE         +12c  1  B  action response from user on HardErr
TCBfIOLock          +12d  1  B  NZ if TCBLockHndl is valid
TCBLockHndl         +12e  C  S  Lock handle of user mem
TCBThisCDS          +13a  4  D  Address of current CDS
TCBThisFSC          +13e  4  D  address of current FSC
TCBpTmpCDS          +142  4  D  Address of dummycds
TCBpOpenBuf         +146  2  W  Address of current OpenBuf
TCBpSearchBuf       +148  2  W  Address of SearchBuf
TCBFailErr          +14a  2  W  NZ if user did FAIL on I 24
TCB_SemInfo         +14c  4  D  16bit addr of the ramsem blocked upon
TCB_SemDebugAddr    +150  4  D  debugger display address for ksems
TCB_NPX_Buffer      +154  4  D
TCBpTCBWaitNext     +158  4  D  Next waiting TCB
TCBpTCBWaitList     +15c  4  D  Threads waiting for me to die
TCBQState           +160  1  B  Scheduler queue location (actual)
TCBState            +161  1  B  Current scheduler state (desired)
TCBWakeFlags        +162  1  B  TKSleep/TKWakeup Flags
TCBcWindowBoost     +163  1  B  Window Boost count
TCBPriClass         +164  1  B  Priority Class (user)
TCBPriLevel         +165  1  B  Priority Level (user)
TCBPriClassMod      +166  1  B  Priority Class modifier bits
TCBSchFlags         +167  1  B  Misc. Scheduler flags
TCBPriority         +168  2  W  Calculated Priority
TCBPriorityMin      +16a  2  W  Minimum Scheduling priority
TCBcBoostLock +16c 4 D Kernel Boost Lock nesting count.
TCBpTCBpPriNextQ +170 4 D Next priority queue in chain
TCBpTCBpPriPrevQ +174 4 D Previous priority queue in chain
TCBpTCBpPriHigher +178 4 D Higher priority thread
TCBpTCBpPriLower +17c 4 D Lower priority thread
TCBpTCBpPriNext +180 4 D Next same-priority thread
TCBpTCBpPriPrev +184 4 D Prev same-priority thread
TCBpTCBWakeup +188 4 D TKQueryWakeup TCB list
TCBSleepID +18c 4 D Sleep ID this TCB is sleeping on
TCBtoe +190 10 S Timeout/Starvation Timeout element
TCBCheckedSig +1a0 1 B Used by the loader
TCBSwapping +1a1 1 B status of swapping
TCBVolIONest +1a2 1 B nesting level of FSH_DoVolIO
TCBReqPktFlg +1a3 1 B Flag to indicate if request pkt in use
TCBReqPkt +1a4 4 D I/O request packet for thread
TCBSysTime +1a8 4 D time spent in system code
TCBUserTime +1ac 4 D time spent in user code
TCB_pPVDBThd +1b0 4 D Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_fIDbg +1b4 4 D
TCBpCpl2_ESP +1b8 4 D Saved TSS CPL2 stack pointer.
TCBpCpl2_SS +1bc 2 W Saved TSS CPL2 stack segment.
TCBNewFlags +1be 2 B Value copied from ptda_NewFiles
TCBEntryActions +1bf 1 B Kernel entry force flags
TCBSig_pend +1c0 2 W bit vector of pending signals
TCBSig_holding +1c2 2 W bit vector of postponed signals
TCBSig_cur +1c4 2 W bit vec of signals being processed
TCBXcptRepRec +1c6 4 D report record of active exception
TCBSig_termtid +1ca 2 W tid of terminator -75797
TCBSecbits +1cc 1 B Security bits 54735
TCBspbytes +1cd 1 B To keep size 4*N 54735
TCB_ulSRIndex +1ce 4 D Last semaphore cleared in MUX 72485
TCBMiscFlags +1d2 1 B Used for hard error processing
TCBModeFlags +1d3 2 W Mode flags for OPEN - for WhatVolume
TCBSpareFlags +1d5 1 B Spare flags
TCBLibiFlags +1d6 1 B 84537
TCBFiller +1d7 1 B To keep size 4*N
TCB_ProcNameBuf +1d8 4 D Pointer to procedure name
TCB_ObjNameBuf +1dc 4 D Pointer to object name buffer
TCB_TmpNameBuf +1e0 4 D aka TCB_TgtModNameBuf
Thread Control Block for OS/2 Warp V3.0 with Fix-Pack

See: Fix pack 09 for details of the change introduced in this fix-pack.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBOrdinal</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Ordinal number of thread in PTDA</td>
</tr>
<tr>
<td>TCBNumber</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread slot number</td>
</tr>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBptib</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>forward link to next (active) TCB</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBpStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBpStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBpKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBpKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBpKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBpKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBpKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
<tr>
<td>Field Name</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>TCBpKStackPTDA</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>KStack page 2 of PTDA</td>
</tr>
<tr>
<td>TCBCurTCB</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TCB</td>
</tr>
<tr>
<td>TCBCurTSD</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TSD</td>
</tr>
<tr>
<td>TCBBiasTCB</td>
<td>+6c</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TCB conversion value</td>
</tr>
<tr>
<td>TCBBiasTSD</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TSD conversion value</td>
</tr>
<tr>
<td>TCBpDHRetAddr</td>
<td>+74</td>
<td>4</td>
<td>D</td>
<td>$2818 Pointer to DHRouter return address</td>
</tr>
<tr>
<td>TCBTLMA</td>
<td>+78</td>
<td>80</td>
<td>D</td>
<td>Thread local memory area</td>
</tr>
<tr>
<td>TCBDMAAdd</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>Position of first sector accessed within file</td>
</tr>
<tr>
<td>TCBThisSFT</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>pointer to SFT we're working with</td>
</tr>
<tr>
<td>TCBValSec</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Number of valid (previously written) sectors</td>
</tr>
<tr>
<td>TCBpRTCB</td>
<td>+108</td>
<td>4</td>
<td>D</td>
<td>Redirector TCB (Used by LANMAN)</td>
</tr>
<tr>
<td>TCBProc_ID</td>
<td>+10c</td>
<td>2</td>
<td>W</td>
<td>process ID for file sharing checks</td>
</tr>
<tr>
<td>TCBUser_ID</td>
<td>+10e</td>
<td>2</td>
<td>W</td>
<td>user ID for file sharing checks</td>
</tr>
<tr>
<td>TCBfSharing</td>
<td>+110</td>
<td>1</td>
<td>B</td>
<td>non-zero ==&gt; no redirection</td>
</tr>
<tr>
<td>TCBfSharing</td>
<td>+111</td>
<td>1</td>
<td>B</td>
<td>see SetAttrib/file.asm</td>
</tr>
<tr>
<td>TCBfIOLock</td>
<td>+112</td>
<td>1</td>
<td>B</td>
<td>JFN flag bits for current fil handle</td>
</tr>
<tr>
<td>TCBAIAllowed</td>
<td>+113</td>
<td>1</td>
<td>B</td>
<td>Allowed I 24 answers (see allowed_)</td>
</tr>
<tr>
<td>TCBopCookie</td>
<td>+114</td>
<td>4</td>
<td>D</td>
<td>server's per file cookie</td>
</tr>
<tr>
<td>TCBopFlags</td>
<td>+118</td>
<td>2</td>
<td>W</td>
<td>whether server wants oplock, etc.</td>
</tr>
<tr>
<td>TCBCurBuf</td>
<td>+11a</td>
<td>4</td>
<td>D</td>
<td>currently assigned buffer</td>
</tr>
<tr>
<td>TCBThisVPB</td>
<td>+11e</td>
<td>2</td>
<td>W</td>
<td>handle of current VPB</td>
</tr>
<tr>
<td>TCBBetNextAdd</td>
<td>+120</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBThisSecPos</td>
<td>+122</td>
<td>2</td>
<td>W</td>
<td>position of first byte within sector</td>
</tr>
<tr>
<td>TCBClusNum</td>
<td>+124</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBLastPosX</td>
<td>+126</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytCnt1</td>
<td>+128</td>
<td>2</td>
<td>W</td>
<td>Number of bytes in 1st sector</td>
</tr>
<tr>
<td>TCBBytCnt2</td>
<td>+12a</td>
<td>2</td>
<td>W</td>
<td># of bytes in last sector</td>
</tr>
<tr>
<td>TCBSecCnt</td>
<td>+12c</td>
<td>2</td>
<td>W</td>
<td>number of whole sectors</td>
</tr>
<tr>
<td>TCBSecClusPos</td>
<td>+12e</td>
<td>1</td>
<td>B</td>
<td>posit of first sector within cluster</td>
</tr>
<tr>
<td>TCBBufHE</td>
<td>+12f</td>
<td>1</td>
<td>B</td>
<td>How to handle a HardError</td>
</tr>
<tr>
<td>TCBactBufHE</td>
<td>+130</td>
<td>1</td>
<td>B</td>
<td>action response from user on HardErr</td>
</tr>
<tr>
<td>TCBfIOLock</td>
<td>+131</td>
<td>1</td>
<td>B</td>
<td>NZ if TCBLockHndl is valid</td>
</tr>
<tr>
<td>TCBLockHndl</td>
<td>+132</td>
<td>2</td>
<td>C</td>
<td>Lock handle of user mem</td>
</tr>
<tr>
<td>TCBThisCDS</td>
<td>+13e</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS</td>
</tr>
<tr>
<td>TCBThisFSC</td>
<td>+142</td>
<td>4</td>
<td>D</td>
<td>address of current FSC</td>
</tr>
<tr>
<td>TCBpTmpCDS</td>
<td>+146</td>
<td>4</td>
<td>D</td>
<td>Address of dummycds</td>
</tr>
<tr>
<td>TCBpOpenBuf</td>
<td>+14a</td>
<td>2</td>
<td>W</td>
<td>Address of current OpenBuf</td>
</tr>
<tr>
<td>TCBpSearchBuf</td>
<td>+14c</td>
<td>2</td>
<td>W</td>
<td>Address of SearchBuf</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCBFailErr</td>
<td>+14e</td>
<td>2</td>
<td>W</td>
<td>NZ if user did FAIL on I 24</td>
</tr>
<tr>
<td>TCB_SemInfo</td>
<td>+150</td>
<td>4</td>
<td>D</td>
<td>16-bit addr of the ramsem blocked upon</td>
</tr>
<tr>
<td>TCB_SemDebugAddr</td>
<td>+154</td>
<td>4</td>
<td>D</td>
<td>Debugger display address for ksems</td>
</tr>
<tr>
<td>TCB_NPX_Buffer</td>
<td>+158</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBpTCBWaitNext</td>
<td>+15c</td>
<td>4</td>
<td>D</td>
<td>Next waiting TCB</td>
</tr>
<tr>
<td>TCBpTCBWaitList</td>
<td>+160</td>
<td>4</td>
<td>D</td>
<td>Threads waiting for me to die</td>
</tr>
<tr>
<td>TCBQState</td>
<td>+164</td>
<td>1</td>
<td>B</td>
<td>Scheduler queue location (actual)</td>
</tr>
<tr>
<td>TCBState</td>
<td>+165</td>
<td>1</td>
<td>B</td>
<td>Current scheduler state (desired)</td>
</tr>
<tr>
<td>TCBWakeFlags</td>
<td>+166</td>
<td>1</td>
<td>B</td>
<td>TKSleep/TKWakeup Flags</td>
</tr>
<tr>
<td>TCBcWindowBoost</td>
<td>+167</td>
<td>1</td>
<td>B</td>
<td>Window Boost count</td>
</tr>
<tr>
<td>TCBpTCBClass</td>
<td>+168</td>
<td>1</td>
<td>B</td>
<td>Priority Class (user)</td>
</tr>
<tr>
<td>TCBPriLevel</td>
<td>+169</td>
<td>1</td>
<td>B</td>
<td>Priority Level (user)</td>
</tr>
<tr>
<td>TCBPriClassMod</td>
<td>+16a</td>
<td>1</td>
<td>B</td>
<td>Priority Class modifier bits</td>
</tr>
<tr>
<td>TCBSchFlags</td>
<td>+16b</td>
<td>1</td>
<td>B</td>
<td>Misc. Scheduler flags</td>
</tr>
<tr>
<td>TCBPriority</td>
<td>+16c</td>
<td>2</td>
<td>W</td>
<td>Calculated Priority</td>
</tr>
<tr>
<td>TCBPriorityMin</td>
<td>+16e</td>
<td>2</td>
<td>W</td>
<td>Minimum Scheduling priority</td>
</tr>
<tr>
<td>TCBcBoostLock</td>
<td>+170</td>
<td>4</td>
<td>D</td>
<td>Kernel Boost Lock nesting count</td>
</tr>
<tr>
<td>TCBpTCBPriNextQ</td>
<td>+174</td>
<td>4</td>
<td>D</td>
<td>Next priority queue in chain</td>
</tr>
<tr>
<td>TCBpTCBPriPrevQ</td>
<td>+178</td>
<td>4</td>
<td>D</td>
<td>Previous priority queue in chain</td>
</tr>
<tr>
<td>TCBpTCBPriHigher</td>
<td>+17c</td>
<td>4</td>
<td>D</td>
<td>Higher priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriLower</td>
<td>+180</td>
<td>4</td>
<td>D</td>
<td>Lower priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriNext</td>
<td>+184</td>
<td>4</td>
<td>D</td>
<td>Next same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriPrev</td>
<td>+188</td>
<td>4</td>
<td>D</td>
<td>Prev same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBWakeups</td>
<td>+18c</td>
<td>4</td>
<td>D</td>
<td>TKQueryWakeup TCB list</td>
</tr>
<tr>
<td>TCBSleepID</td>
<td>+190</td>
<td>4</td>
<td>D</td>
<td>Sleep ID this TCB is sleeping on</td>
</tr>
<tr>
<td>TCBte</td>
<td>+194</td>
<td>10</td>
<td>S</td>
<td>Timeout/Starvation Timeout element</td>
</tr>
<tr>
<td>TCBCheckedSig</td>
<td>+1a4</td>
<td>1</td>
<td>B</td>
<td>Used by the loader</td>
</tr>
<tr>
<td>TCBfSwapping</td>
<td>+1a5</td>
<td>1</td>
<td>B</td>
<td>Status of swapping</td>
</tr>
<tr>
<td>TCBVolIONest</td>
<td>+1a6</td>
<td>1</td>
<td>B</td>
<td>Nesting level of FSR_DoVolIO</td>
</tr>
<tr>
<td>TCBReqPktFlg</td>
<td>+1a7</td>
<td>1</td>
<td>B</td>
<td>Flag to indicate if request pkt in use</td>
</tr>
<tr>
<td>TCBReqPkt</td>
<td>+1a8</td>
<td>4</td>
<td>D</td>
<td>I/O request packet for thread</td>
</tr>
<tr>
<td>TCBSysTime</td>
<td>+1ac</td>
<td>4</td>
<td>D</td>
<td>Time spent in system code</td>
</tr>
<tr>
<td>TCBUserTime</td>
<td>+1b0</td>
<td>4</td>
<td>D</td>
<td>Time spent in user code</td>
</tr>
<tr>
<td>TCB_pPVDBThd</td>
<td>+1b4</td>
<td>4</td>
<td>D</td>
<td>Pointer to Perfview Data Block for this thread (pvdb_thd_s).</td>
</tr>
<tr>
<td>TCB_f1Dbg</td>
<td>+1b8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBpCl2_ESP</td>
<td>+1bc</td>
<td>4</td>
<td>D</td>
<td>Saved TSS CPL2 stack pointer.</td>
</tr>
<tr>
<td>TCBpCl2_SS</td>
<td>+1c0</td>
<td>2</td>
<td>W</td>
<td>Saved TSS CPL2 stack segment.</td>
</tr>
<tr>
<td>TCBNewFlags</td>
<td>+1c2</td>
<td>1</td>
<td>B</td>
<td>Value copied from ptda_NewFiles</td>
</tr>
<tr>
<td>TCBEntryActions</td>
<td>+1c3</td>
<td>1</td>
<td>B</td>
<td>Kernel entry force flags</td>
</tr>
</tbody>
</table>
Thread Control Block for OS/2 Warp V3.0 with Fix-Pack 11 or Later

Field Name | Offset | Length | Type | Description
--- | --- | --- | --- | ---
TCBOrdinal | +0 | 2 | W | Ordinal number of thread in PTDA
TCBNumber | +2 | 2 | W | Thread slot number
<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBptib</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>forward link to next (active) TCB</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBpStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBpStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBpteKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBpteKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA2</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>KStack page 2 of PTDA</td>
</tr>
<tr>
<td>TCBcurrTCB</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TCB</td>
</tr>
<tr>
<td>TCBcurrTSD</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TSD</td>
</tr>
<tr>
<td>TCBBiasTCB</td>
<td>+6c</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TCB conversion value</td>
</tr>
<tr>
<td>TCBBiasTSD</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TSD conversion value</td>
</tr>
<tr>
<td>TCBpDHRetAddr</td>
<td>+74</td>
<td>4</td>
<td>D</td>
<td>82818 Pointer to DHRouter return address</td>
</tr>
<tr>
<td>TCBTLMA</td>
<td>+78</td>
<td>80</td>
<td>D</td>
<td>Thread local memory area</td>
</tr>
<tr>
<td>TCBDMAAdd</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>Position of first sector accessed within file</td>
</tr>
<tr>
<td>TCBThisSFT</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>pointer to SFT we're working with</td>
</tr>
<tr>
<td>TCBValSec</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Number of valid (previously written) sectors</td>
</tr>
<tr>
<td>TCBpRTCBI</td>
<td>+108</td>
<td>4</td>
<td>D</td>
<td>Redirector TCB (Used by LANMAN)</td>
</tr>
<tr>
<td>TCBProc_ID</td>
<td>+10c</td>
<td>2</td>
<td>W</td>
<td>process ID for file sharing checks</td>
</tr>
<tr>
<td>TCBUser_ID</td>
<td>+10e</td>
<td>2</td>
<td>W</td>
<td>user ID for file sharing checks</td>
</tr>
<tr>
<td>TCBfSharing</td>
<td>+110</td>
<td>1</td>
<td>B</td>
<td>non-zero =&gt; no redirection</td>
</tr>
<tr>
<td>TCBsrvAttrib</td>
<td>+111</td>
<td>1</td>
<td>B</td>
<td>see SetAttrib/file.asm</td>
</tr>
<tr>
<td>TCBJfnFlag</td>
<td>+112</td>
<td>1</td>
<td>B</td>
<td>JFN flag bits for current fil handle</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>TCBAllowed</td>
<td>+113</td>
<td>1</td>
<td>B</td>
<td>Allowed I 24 answers (see allowed_)</td>
</tr>
<tr>
<td>TCBOpCookie</td>
<td>+114</td>
<td>4</td>
<td>D</td>
<td>server's per file cookie</td>
</tr>
<tr>
<td>TCBOpFlags</td>
<td>+118</td>
<td>2</td>
<td>W</td>
<td>whether server wants oplock, etc.</td>
</tr>
<tr>
<td>TCBCurBuf</td>
<td>+11a</td>
<td>4</td>
<td>D</td>
<td>currently assigned buffer</td>
</tr>
<tr>
<td>TCBThishVPB</td>
<td>+11e</td>
<td>2</td>
<td>W</td>
<td>handle of current VFB</td>
</tr>
<tr>
<td>TCBNextAdd</td>
<td>+120</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytSecPos</td>
<td>+122</td>
<td>2</td>
<td>W</td>
<td>position of first byte within sector</td>
</tr>
<tr>
<td>TCBClusNum</td>
<td>+124</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBLastPos</td>
<td>+126</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytCnt1</td>
<td>+128</td>
<td>2</td>
<td>W</td>
<td>Number of bytes in 1st sector</td>
</tr>
<tr>
<td>TCBBytCnt2</td>
<td>+12a</td>
<td>2</td>
<td>W</td>
<td># of bytes in last sector</td>
</tr>
<tr>
<td>TCBSecCnt</td>
<td>+12c</td>
<td>2</td>
<td>W</td>
<td>number of whole sectors</td>
</tr>
<tr>
<td>TCBSecClusPos</td>
<td>+12e</td>
<td>1</td>
<td>B</td>
<td>posit of first sector within cluster</td>
</tr>
<tr>
<td>TCBBufHE</td>
<td>+12f</td>
<td>1</td>
<td>B</td>
<td>How to handle a HardError</td>
</tr>
<tr>
<td>TCBactBufHE</td>
<td>+130</td>
<td>1</td>
<td>B</td>
<td>action response from user on HardErr</td>
</tr>
<tr>
<td>TCBfIOLock</td>
<td>+131</td>
<td>1</td>
<td>B</td>
<td>NZ if TCBLockHndl is valid</td>
</tr>
<tr>
<td>TCBLockHndl</td>
<td>+132</td>
<td>C</td>
<td>S</td>
<td>Lock handle of user mem</td>
</tr>
<tr>
<td>TCBThisCDS</td>
<td>+13e</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS</td>
</tr>
<tr>
<td>TCBThisFSC</td>
<td>+142</td>
<td>4</td>
<td>D</td>
<td>address of current FSC</td>
</tr>
<tr>
<td>TCBpTmpCDS</td>
<td>+146</td>
<td>4</td>
<td>D</td>
<td>Address of dummycds</td>
</tr>
<tr>
<td>TCBpOpenBuf</td>
<td>+14a</td>
<td>2</td>
<td>W</td>
<td>Address of current OpenBuf</td>
</tr>
<tr>
<td>TCBpSearchBuf</td>
<td>+14c</td>
<td>2</td>
<td>W</td>
<td>Address of SearchBuf</td>
</tr>
<tr>
<td>TCBFailErr</td>
<td>+14e</td>
<td>2</td>
<td>W</td>
<td>NZ if user did FAIL on I 24</td>
</tr>
<tr>
<td>TCB_SemInfo</td>
<td>+150</td>
<td>4</td>
<td>D</td>
<td>16bit addr of the ramsem blocked upon</td>
</tr>
<tr>
<td>TCB_SemDebugAddr</td>
<td>+154</td>
<td>4</td>
<td>D</td>
<td>debugger display address for ksems</td>
</tr>
<tr>
<td>TCB_NPX_Buffer</td>
<td>+158</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBpTCBWaitNext</td>
<td>+15c</td>
<td>4</td>
<td>D</td>
<td>Next waiting TCB</td>
</tr>
<tr>
<td>TCBpTCBWaitList</td>
<td>+160</td>
<td>4</td>
<td>D</td>
<td>Threads waiting for me to die</td>
</tr>
<tr>
<td>TCBQState</td>
<td>+164</td>
<td>1</td>
<td>B</td>
<td>Scheduler queue location (actual)</td>
</tr>
<tr>
<td>TCBstate</td>
<td>+165</td>
<td>1</td>
<td>B</td>
<td>Current scheduler state (desired)</td>
</tr>
<tr>
<td>TCBWakeFlags</td>
<td>+166</td>
<td>1</td>
<td>B</td>
<td>TKSleep/TKWakeup Flags</td>
</tr>
<tr>
<td>TCBCWindowFlags</td>
<td>+167</td>
<td>1</td>
<td>B</td>
<td>Window Boost count</td>
</tr>
<tr>
<td>TCBPriClass</td>
<td>+168</td>
<td>1</td>
<td>B</td>
<td>Priority Class (user)</td>
</tr>
<tr>
<td>TCBPriLevel</td>
<td>+169</td>
<td>1</td>
<td>B</td>
<td>Priority Level (user)</td>
</tr>
<tr>
<td>TCBPriClassMod</td>
<td>+16a</td>
<td>1</td>
<td>B</td>
<td>Priority Class modifier bits</td>
</tr>
<tr>
<td>TCBSchFlags</td>
<td>+16b</td>
<td>1</td>
<td>B</td>
<td>Misc. Scheduler flags</td>
</tr>
<tr>
<td>TCBPriority</td>
<td>+16c</td>
<td>2</td>
<td>W</td>
<td>Calculated Priority</td>
</tr>
<tr>
<td>TCBPriorityMin</td>
<td>+16e</td>
<td>2</td>
<td>W</td>
<td>Minimum Scheduling priority</td>
</tr>
<tr>
<td>TCBCBoostLock</td>
<td>+170</td>
<td>4</td>
<td>D</td>
<td>Kernel Boost Lock nesting count.</td>
</tr>
</tbody>
</table>
TCBpTCBPriNextQ +174 4 D Next priority queue in chain
TCBpTCBPriPrevQ +178 4 D Previous priority queue in chain
TCBpTCBPriHigher +17c 4 D Higher priority thread
TCBpTCBPriLower +180 4 D Lower priority thread
TCBpTCBPriNext +184 4 D Next same-priority thread
TCBpTCBPriPrev +188 4 D Prev same-priority thread
TCBpTCBWakeup +18c 4 D TKQueryWakeup TCB list
TCBSleepID +190 4 D Sleep ID this TCB is sleeping on
TCBtoe +194 14 S Timeout/Starvation Timeout element
TCBCheckedSig +1a8 1 B Used by the loader
TCBfSwapping +1a9 1 B status of swapping
TCBVc1IONest +1aa 1 B nesting level of FSR_DoVolIO
TCBReqPktFlg +1ab 1 B Flag to indicate if request pkt in use
TCBReqPkt +1ac 4 D I/O request packet for thread
TCBSysTime +1b0 4 D time spent in system code
TCBUserTime +1b4 4 D time spent in user code
TCB_pPVDBThd +1b8 4 D Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_f1Dbg +1bc 4 D
TCBcpl2_ESP +1c0 4 D Saved TSS CPL2 stack pointer.
TCBcpl2_SS +1c4 2 W Saved TSS CPL2 stack segment.
TCBNewFlags +1c6 1 B Value copied from ptda_NewFiles
TCBEntryActions +1c7 1 B Kernel entry force flags
TCBSig_pend +1c8 2 W bit vector of pending signals
TCBSig_holding +1ca 2 W bit vector of postponed signals
TCBSig_cur +1cc 2 W bit vec of signals being processed
TCBXceptRepRec +1ce 4 D report record of active exception
TCBSig_termtid +1d2 2 W tid of terminator -75797
TCBSecbits +1d4 1 B Security bits 54735
TCBspbytes +1d5 1 B To keep size 4*N 54735
TCB_u1SRIndex +1d6 4 D Last semaphore cleared in MUX 72485
TCBMiscFlags +1da 1 B Used for hard error processing
TCBModeFlags +1db 2 W Mode flags for OPEN - for WhatVolume
TCBSpareFlags +1dd 1 B Spare flags
TCBLibFlags +1de 1 B 84537
TCBFiller +1df 1 B To keep size 4*N
TCB_ProcNameBuf +1e0 4 D Pointer to procedure name
TCB_ObjNameBuf +1e4 4 D Pointer to object name buffer
TCB_TmpNameBuf +1e8 4 D aka TCB_TgtModNameBuf
TCB_SrcModNameBuf +1ec 4 D Used by loader
Thread Control Block for OS/2 V2.11 with Fix-Pack 90 or Later

See: Fix pack 90 for details of the change introduced in this fix-pack.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBOrdinal</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Ordinal number of thread in PTDA</td>
</tr>
<tr>
<td>TCBNumber</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread slot number</td>
</tr>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBptib</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>forward link to next (active) TCB</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBpStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBpStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBppteKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBppteKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBppteKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBppteKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBppteKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
</tbody>
</table>
TCBpteKStackPTDA2 +60  4  D  KStack page 2 of PTDA
TCBCurrTCB +64  4  D  SS-relative offset of Current TCB
TCBCurrTSD +68  4  D  SS-relative offset of Current TSD
TCBBiasTCB +6c  4  D  stack-to-flat TCB conversion value
TCBBiasTSD +70  4  D  stack-to-flat TSD conversion value
TCBPDHRetAddr +74  4  D  $8218 Pointer to DHRouter return address
TCBDMAAddr +78  4  D  User's I/O transfer address
TCBSecPos +7c  4  D  Position of first sector accessed within file
TCBThisSFT +80  4  D  pointer to SFT we're working with
TCBValSec +84  4  D  Number of valid (previously written) sectors
TCBProc_ID +88  2  W  process ID for file sharing checks
TCBUser_ID +8c  2  W  user ID for file sharing checks
TCBfSharing +90  1  B  non-zero ==> no redirection
TCBSrvAttrib +91  1  B  see SetAttrib/file.asm
TCBJfnFlag +92  1  B  JFN flag bits for current file handle
TCBAllowed +93  1  B  Allowed I 24 answers (see allowed_)
TCBOpCookie +94  4  D  server's per file cookie
TCBOpFlags +98  2  W  whether server wants oplock, etc.
TCBCurBuf +9a  4  D  currently assigned buffer
TCBThisVPB +9e  2  W  handle of current VPB
TCBNextAdd +a0  2  W
TCBBytSecPos +a2  2  W  position of first byte within sector
TCBClusNum +a4  2  W
TCBLastPos +a6  2  W
TCBBytCnt1 +a8  2  W  Number of bytes in 1st sector
TCBBytCnt2 +aa  2  W  # of bytes in last sector
TCBSecCnt +ac  2  W  number of whole sectors
TCBSecClusPos +ae  1  B  posit of first sector within cluster
TCBufHE +af  1  B  How to handle a HardError
TCBactBufHE +b0  1  B  action response from user on HardErr
TCBfIOLock +b1  1  B  NZ if TCBLockHndl is valid
TCBLockHndl +b2  2  C  S  Lock handle of user mem
TCBThisCDS +be  4  D  Address of current CDS
TCBThisFSC +c2  4  D  address of current FSC
TCBpTmpCDS +c6  4  D  Address of dummycds
TCBpOpenBuf +ca  2  W  Address of current OpenBuf
TCBpSearchBuf +cc  2  W  Address of SearchBuf
TCBFailErr +ce  2  W  NZ if user did FAIL on I 24
<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCB_SemInfo</td>
<td>+d0</td>
<td>4</td>
<td>D</td>
<td>16bit addr of the ramsem blocked upon</td>
</tr>
<tr>
<td>TCB_SemDebugAddr</td>
<td>+d4</td>
<td>4</td>
<td>D</td>
<td>debugger display address for ksems</td>
</tr>
<tr>
<td>TCB_NPX_Buffer</td>
<td>+d8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBpTCBWaitNext</td>
<td>+dc</td>
<td>4</td>
<td>D</td>
<td>Next waiting TCB</td>
</tr>
<tr>
<td>TCBpTCBWaitList</td>
<td>+e0</td>
<td>4</td>
<td>D</td>
<td>Threads waiting for me to die</td>
</tr>
<tr>
<td>TCBState</td>
<td>+e4</td>
<td>1</td>
<td>B</td>
<td>Scheduler queue location (actual)</td>
</tr>
<tr>
<td>TCBState</td>
<td>+e5</td>
<td>1</td>
<td>B</td>
<td>Current scheduler state (desired)</td>
</tr>
<tr>
<td>TCBWakeFlags</td>
<td>+e6</td>
<td>1</td>
<td>B</td>
<td>TKSleep/TKWakeup Flags</td>
</tr>
<tr>
<td>TCBcWindowBoost</td>
<td>+e7</td>
<td>1</td>
<td>B</td>
<td>Window Boost count</td>
</tr>
<tr>
<td>TCBpPriClass</td>
<td>+e8</td>
<td>1</td>
<td>B</td>
<td>Priority Class (user)</td>
</tr>
<tr>
<td>TCBpPriLevel</td>
<td>+e9</td>
<td>1</td>
<td>B</td>
<td>Priority Level (user)</td>
</tr>
<tr>
<td>TCBpPriClassMod</td>
<td>+ea</td>
<td>1</td>
<td>B</td>
<td>Priority Class modifier bits</td>
</tr>
<tr>
<td>TCB_SchFlags</td>
<td>+eb</td>
<td>1</td>
<td>B</td>
<td>Misc. Scheduler flags</td>
</tr>
<tr>
<td>TCBPriority</td>
<td>+ec</td>
<td>2</td>
<td>W</td>
<td>Calculated Priority</td>
</tr>
<tr>
<td>TCBPriorityMin</td>
<td>+ee</td>
<td>2</td>
<td>W</td>
<td>Minimum Scheduling priority</td>
</tr>
<tr>
<td>TCBcBoostLock</td>
<td>+f0</td>
<td>4</td>
<td>D</td>
<td>Kernel Boost Lock nesting count</td>
</tr>
<tr>
<td>TCBpTCBPriNextQ</td>
<td>+f4</td>
<td>4</td>
<td>D</td>
<td>Next priority queue in chain</td>
</tr>
<tr>
<td>TCBpTCBPriPrevQ</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>Previous priority queue in chain</td>
</tr>
<tr>
<td>TCBpTCBPriHigher</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>Higher priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriLower</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>Lower priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriNext</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Next same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriPrev</td>
<td>+108</td>
<td>4</td>
<td>D</td>
<td>Prev same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBWake</td>
<td>+10c</td>
<td>4</td>
<td>D</td>
<td>TKQueryWakeup TCB list</td>
</tr>
<tr>
<td>TCBSleepID</td>
<td>+110</td>
<td>4</td>
<td>D</td>
<td>Sleep ID this TCB is sleeping on</td>
</tr>
<tr>
<td>TCBtoe</td>
<td>+114</td>
<td>14</td>
<td>S</td>
<td>Timeout/Starvation Timeout element</td>
</tr>
<tr>
<td>TCBCheckedSig</td>
<td>+128</td>
<td>1</td>
<td>B</td>
<td>Used by the loader</td>
</tr>
<tr>
<td>TCBfSwapping</td>
<td>+129</td>
<td>1</td>
<td>B</td>
<td>status of swapping</td>
</tr>
<tr>
<td>TCBVoiIONest</td>
<td>+12a</td>
<td>1</td>
<td>B</td>
<td>nesting level of FSR_DoVolIO</td>
</tr>
<tr>
<td>TCBReqPktFlg</td>
<td>+12b</td>
<td>1</td>
<td>B</td>
<td>Flag to indicate if request pkt in use</td>
</tr>
<tr>
<td>TCBReqPkt</td>
<td>+12c</td>
<td>4</td>
<td>D</td>
<td>I/O request packet for thread</td>
</tr>
<tr>
<td>TCBpMemStatCur</td>
<td>+130</td>
<td>4</td>
<td>D</td>
<td>Current structure being filled in</td>
</tr>
<tr>
<td>TCBMemStat</td>
<td>+134</td>
<td>3C</td>
<td>S</td>
<td>statistics structure</td>
</tr>
<tr>
<td>TCBSysTime</td>
<td>+170</td>
<td>4</td>
<td>D</td>
<td>time spent in system code</td>
</tr>
<tr>
<td>TCBUserTime</td>
<td>+174</td>
<td>4</td>
<td>D</td>
<td>time spent in user code</td>
</tr>
<tr>
<td>TCB_pPVDBThd</td>
<td>+178</td>
<td>4</td>
<td>D</td>
<td>Ptr to Perfview Data Block for this thread (pvdb_thd_s)</td>
</tr>
<tr>
<td>TCB_f10dbg</td>
<td>+17c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBPciL2_ESP</td>
<td>+180</td>
<td>4</td>
<td>D</td>
<td>Saved TSS CPL2 stack pointer.</td>
</tr>
<tr>
<td>TCBPciL2_SS</td>
<td>+184</td>
<td>2</td>
<td>W</td>
<td>Saved TSS CPL2 stack segment.</td>
</tr>
<tr>
<td>TCBNewFlags</td>
<td>+186</td>
<td>1</td>
<td>W</td>
<td>Value copied from ptda_NewFiles</td>
</tr>
</tbody>
</table>
Thread Control Block for OS/2 V2.11

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBOrdinal</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Ordinal number of thread in PTDA</td>
</tr>
<tr>
<td>TCBNumber</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Thread slot number</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>TCBForcedActions</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Bit vector of forced actions</td>
</tr>
<tr>
<td>TCBpPTDA</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Pointer to the PTDA</td>
</tr>
<tr>
<td>TCBpTSD</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread swappable data</td>
</tr>
<tr>
<td>TCBptib</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Pointer to thread info block</td>
</tr>
<tr>
<td>TCBpTCBNext</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>forward link to next (active) TCB</td>
</tr>
<tr>
<td>TCBcbStackMax</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Virtual size of stack object</td>
</tr>
<tr>
<td>TCBcbStackCur</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Committed size of stack object</td>
</tr>
<tr>
<td>TCBStack</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Virtual base of stack</td>
</tr>
<tr>
<td>TCBpStack16Lo</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Virtual base of 16-bit stack</td>
</tr>
<tr>
<td>TCBStack16Hi</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Virtual limit of 16-bit stack</td>
</tr>
<tr>
<td>TCBpLibiHead</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiCurr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Link to libi load data area</td>
</tr>
<tr>
<td>TCBpLibiFree</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Link to libi free data area</td>
</tr>
<tr>
<td>TCB_pcriFrameType</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>stack frame type</td>
</tr>
<tr>
<td>TCB_pFrameBase</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>stack frame base pointer</td>
</tr>
<tr>
<td>TCB_hookheadLocal</td>
<td>+40</td>
<td>8</td>
<td>D</td>
<td>local context hook head</td>
</tr>
<tr>
<td>TCB_phookOwnerHead</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>linked list of hook blocks</td>
</tr>
<tr>
<td>TCBpteKStackTCB0</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTCB1</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of TCB</td>
</tr>
<tr>
<td>TCBpteKStackTSD</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>KStack TSD page</td>
</tr>
<tr>
<td>TCBpteKStackPTDA0</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>KStack page 0 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA1</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>KStack page 1 of PTDA</td>
</tr>
<tr>
<td>TCBpteKStackPTDA2</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>KStack page 2 of PTDA</td>
</tr>
<tr>
<td>TCBCurrTCB</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TCB</td>
</tr>
<tr>
<td>TCBCurrTSD</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>SS-relative offset of Current TSD</td>
</tr>
<tr>
<td>TCBBiasTCB</td>
<td>+6c</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TCB conversion value</td>
</tr>
<tr>
<td>TCBBiasTSD</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>stack-to-flat TSD conversion value</td>
</tr>
<tr>
<td>TCBDMAAdd</td>
<td>+74</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address</td>
</tr>
<tr>
<td>TCBSecPos</td>
<td>+78</td>
<td>4</td>
<td>D</td>
<td>Position of first sector accessed within file</td>
</tr>
<tr>
<td>TCBThisSFT</td>
<td>+7c</td>
<td>4</td>
<td>D</td>
<td>pointer to SFT we're working with</td>
</tr>
<tr>
<td>TCBValSec</td>
<td>+80</td>
<td>4</td>
<td>D</td>
<td>Number of valid (previously written) sectors</td>
</tr>
<tr>
<td>TCBpRTCBI</td>
<td>+84</td>
<td>4</td>
<td>D</td>
<td>Redirector TCB (Used by LANMAN)</td>
</tr>
<tr>
<td>TCBProc_ID</td>
<td>+88</td>
<td>2</td>
<td>W</td>
<td>process ID for file sharing checks</td>
</tr>
<tr>
<td>TCBUser_ID</td>
<td>+8a</td>
<td>2</td>
<td>W</td>
<td>user ID for file sharing checks</td>
</tr>
<tr>
<td>TCBfSharing</td>
<td>+8c</td>
<td>1</td>
<td>B</td>
<td>non-zero ==&gt; no redirection</td>
</tr>
<tr>
<td>TCBScrsvAttrib</td>
<td>+8d</td>
<td>1</td>
<td>B</td>
<td>see SetAttrib/file.asm</td>
</tr>
<tr>
<td>TCBJfnFlag</td>
<td>+8e</td>
<td>1</td>
<td>B</td>
<td>JFN flag bits for current fil handle</td>
</tr>
<tr>
<td>TCBAllowed</td>
<td>+8f</td>
<td>1</td>
<td>B</td>
<td>Allowed I 24 answers (see allowed_)</td>
</tr>
<tr>
<td>TCBOpCookie</td>
<td>+90</td>
<td>4</td>
<td>D</td>
<td>server's per file cookie</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>TCBOpFlags</td>
<td>+94</td>
<td>2</td>
<td>W</td>
<td>whether server wants oplock, etc.</td>
</tr>
<tr>
<td>TCBCurBuf</td>
<td>+96</td>
<td>4</td>
<td>D</td>
<td>currently assigned buffer</td>
</tr>
<tr>
<td>TCBThishVPB</td>
<td>+9a</td>
<td>2</td>
<td>W</td>
<td>handle of current VFB</td>
</tr>
<tr>
<td>TCBNxtAdd</td>
<td>+9c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytSecPos</td>
<td>+9e</td>
<td>2</td>
<td>W</td>
<td>position of first byte within sector</td>
</tr>
<tr>
<td>TCBC1usNum</td>
<td>+a0</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBLastPos</td>
<td>+a2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCBBytCnt1</td>
<td>+a4</td>
<td>2</td>
<td>W</td>
<td>Number of bytes in 1st sector</td>
</tr>
<tr>
<td>TCBBytCnt2</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td># of bytes in last sector</td>
</tr>
<tr>
<td>TCBSecCnt</td>
<td>+a8</td>
<td>2</td>
<td>W</td>
<td>number of whole sectors</td>
</tr>
<tr>
<td>TCBSecClusPos</td>
<td>+aa</td>
<td>1</td>
<td>B</td>
<td>posit of first sector within cluster</td>
</tr>
<tr>
<td>TCBBufHE</td>
<td>+ab</td>
<td>1</td>
<td>B</td>
<td>How to handle a HardError</td>
</tr>
<tr>
<td>TCBactBufHE</td>
<td>+ac</td>
<td>1</td>
<td>B</td>
<td>action response from user on HardErr</td>
</tr>
<tr>
<td>TCBfIOLock</td>
<td>+ad</td>
<td>1</td>
<td>B</td>
<td>NZ if TCBLockHndl is valid</td>
</tr>
<tr>
<td>TCBLockHndl</td>
<td>+ae</td>
<td>C</td>
<td>S</td>
<td>Lock handle of user mem</td>
</tr>
<tr>
<td>TCBThisCDS</td>
<td>+ba</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS</td>
</tr>
<tr>
<td>TCBThisFSC</td>
<td>+be</td>
<td>4</td>
<td>D</td>
<td>address of current FSC</td>
</tr>
<tr>
<td>TCBpTmpCDS</td>
<td>+c2</td>
<td>4</td>
<td>D</td>
<td>Address of dummycds</td>
</tr>
<tr>
<td>TCBpOpenBuf</td>
<td>+c6</td>
<td>2</td>
<td>W</td>
<td>Address of current OpenBuf</td>
</tr>
<tr>
<td>TCBpSearchBuf</td>
<td>+c8</td>
<td>2</td>
<td>W</td>
<td>Address of SearchBuf</td>
</tr>
<tr>
<td>TCBFailErr</td>
<td>+ca</td>
<td>2</td>
<td>W</td>
<td>NZ if user did FAIL on I 24</td>
</tr>
<tr>
<td>TCB_SemInfo</td>
<td>+cc</td>
<td>4</td>
<td>D</td>
<td>16bit addr of the ramsem blocked upon</td>
</tr>
<tr>
<td>TCB_SemDebugAddr</td>
<td>+d0</td>
<td>4</td>
<td>D</td>
<td>debugger display address for ksems</td>
</tr>
<tr>
<td>TCB_NPX_Buffer</td>
<td>+d4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCBpTCBWaitNext</td>
<td>+d8</td>
<td>4</td>
<td>D</td>
<td>Next waiting TCB</td>
</tr>
<tr>
<td>TCBpTCBWaitList</td>
<td>+dc</td>
<td>4</td>
<td>D</td>
<td>Threads waiting for me to die</td>
</tr>
<tr>
<td>TCBQState</td>
<td>+e0</td>
<td>1</td>
<td>B</td>
<td>Scheduler queue location (actual)</td>
</tr>
<tr>
<td>TCBState</td>
<td>+e1</td>
<td>1</td>
<td>B</td>
<td>Current scheduler state (desired)</td>
</tr>
<tr>
<td>TCBWakeFlags</td>
<td>+e2</td>
<td>1</td>
<td>B</td>
<td>TKSleep/TKWakeup Flags</td>
</tr>
<tr>
<td>TCBWindowBoost</td>
<td>+e3</td>
<td>1</td>
<td>B</td>
<td>Window Boost count</td>
</tr>
<tr>
<td>TCBPriClass</td>
<td>+e4</td>
<td>1</td>
<td>B</td>
<td>Priority Class (user)</td>
</tr>
<tr>
<td>TCBPriLevel</td>
<td>+e5</td>
<td>1</td>
<td>B</td>
<td>Priority Level (user)</td>
</tr>
<tr>
<td>TCBPriClassMod</td>
<td>+e6</td>
<td>1</td>
<td>B</td>
<td>Priority Class modifier bits</td>
</tr>
<tr>
<td>TCBSchFlags</td>
<td>+e7</td>
<td>1</td>
<td>B</td>
<td>Misc. Scheduler flags</td>
</tr>
<tr>
<td>TCBPriority</td>
<td>+e8</td>
<td>2</td>
<td>W</td>
<td>Calculated Priority</td>
</tr>
<tr>
<td>TCBPriorityMin</td>
<td>+ea</td>
<td>2</td>
<td>W</td>
<td>Minimum Scheduling priority</td>
</tr>
<tr>
<td>TCBootBoostLock</td>
<td>+ec</td>
<td>4</td>
<td>D</td>
<td>Kernel Boost Lock nesting count.</td>
</tr>
<tr>
<td>TCBpTCBPriNextQ</td>
<td>+f0</td>
<td>4</td>
<td>D</td>
<td>Next priority queue in chain</td>
</tr>
<tr>
<td>TCBpTCBPriPrevQ</td>
<td>+f4</td>
<td>4</td>
<td>D</td>
<td>Previous priority queue in chain</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>TCBpTCBPriHigher</td>
<td>+f8</td>
<td>4</td>
<td>D</td>
<td>Higher priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriLower</td>
<td>+fc</td>
<td>4</td>
<td>D</td>
<td>Lower priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriNext</td>
<td>+100</td>
<td>4</td>
<td>D</td>
<td>Next same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBPriPrev</td>
<td>+104</td>
<td>4</td>
<td>D</td>
<td>Prev same-priority thread</td>
</tr>
<tr>
<td>TCBpTCBWakeup</td>
<td>+108</td>
<td>4</td>
<td>D</td>
<td>TKQueryWakeup TCB list</td>
</tr>
<tr>
<td>TCBSleepID</td>
<td>+10c</td>
<td>4</td>
<td>D</td>
<td>Sleep ID this TCB is sleeping on</td>
</tr>
<tr>
<td>TCBtoe</td>
<td>+110</td>
<td>14</td>
<td>S</td>
<td>Timeout/Starvation Timeout element</td>
</tr>
<tr>
<td>TCBCheckedSig</td>
<td>+124</td>
<td>1</td>
<td>B</td>
<td>Used by the loader</td>
</tr>
<tr>
<td>TCBfSwapping</td>
<td>+125</td>
<td>1</td>
<td>B</td>
<td>status of swapping</td>
</tr>
<tr>
<td>TCVolIONext</td>
<td>+126</td>
<td>1</td>
<td>B</td>
<td>nesting level of FSH_DoVolIO</td>
</tr>
<tr>
<td>TCBReqPktFlg</td>
<td>+127</td>
<td>1</td>
<td>B</td>
<td>Flag to indicate if request pkt in use</td>
</tr>
<tr>
<td>TCBReqPkt</td>
<td>+128</td>
<td>4</td>
<td>D</td>
<td>I/O request packet for thread</td>
</tr>
<tr>
<td>TCBpMemStatCur</td>
<td>+12c</td>
<td>4</td>
<td>D</td>
<td>Current structure being filled in</td>
</tr>
<tr>
<td>TCBMemStat</td>
<td>+130</td>
<td>3C</td>
<td>S</td>
<td>statistics structure</td>
</tr>
<tr>
<td>TCBSysTime</td>
<td>+16c</td>
<td>4</td>
<td>D</td>
<td>time spent in system code</td>
</tr>
<tr>
<td>TCBUserTime</td>
<td>+170</td>
<td>4</td>
<td>D</td>
<td>time spent in user code</td>
</tr>
<tr>
<td>TCB_pPVDBThd</td>
<td>+174</td>
<td>4</td>
<td>D</td>
<td>Ptr to Perfview Data Block for this thread (pvdb_thd_s).</td>
</tr>
<tr>
<td>TCB_fIDbg</td>
<td>+178</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCB CPL2 ESP</td>
<td>+17c</td>
<td>4</td>
<td>D</td>
<td>Saved TSS CPL2 stack pointer.</td>
</tr>
<tr>
<td>TCB CPL2 SS</td>
<td>+180</td>
<td>2</td>
<td>W</td>
<td>Saved TSS CPL2 stack segment.</td>
</tr>
<tr>
<td>TCBNewFlags</td>
<td>+182</td>
<td>1</td>
<td>W</td>
<td>Value copied from ptda_NewFiles</td>
</tr>
<tr>
<td>TCB Entry Actions</td>
<td>+183</td>
<td>1</td>
<td>B</td>
<td>Kernel entry force flags</td>
</tr>
<tr>
<td>TCB Sig_pend</td>
<td>+184</td>
<td>2</td>
<td>W</td>
<td>bit vector of pending signals</td>
</tr>
<tr>
<td>TCB Sig_holding</td>
<td>+186</td>
<td>2</td>
<td>W</td>
<td>bit vector of postponed signals</td>
</tr>
<tr>
<td>TCB Sig_cur</td>
<td>+188</td>
<td>2</td>
<td>W</td>
<td>bit vec of signals being processed</td>
</tr>
<tr>
<td>TCB Xcpt Rep Rec</td>
<td>+18a</td>
<td>4</td>
<td>D</td>
<td>report record of active exception</td>
</tr>
<tr>
<td>TCB Sig_termtid</td>
<td>+18e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>TCB Secbits</td>
<td>+190</td>
<td>1</td>
<td>B</td>
<td>Security bits 54735</td>
</tr>
<tr>
<td>TCB spbytes</td>
<td>+191</td>
<td>1</td>
<td>B</td>
<td>To keep size 4*N 54735</td>
</tr>
<tr>
<td>TCB ulSRIndex</td>
<td>+192</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TCB Misc Flags</td>
<td>+196</td>
<td>1</td>
<td>D</td>
<td>Used for hard error processing</td>
</tr>
<tr>
<td>TCB Mode Flags</td>
<td>+197</td>
<td>2</td>
<td>D</td>
<td>Mode flags for OPEN - for WhatVolume</td>
</tr>
<tr>
<td>TCB Spare Flags</td>
<td>+199</td>
<td>1</td>
<td>B</td>
<td>Spare flags</td>
</tr>
<tr>
<td>TCB Libi Flags</td>
<td>+19a</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>TCB Filler</td>
<td>+19b</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>TCB Proc Name Buf</td>
<td>+19c</td>
<td>4</td>
<td>D</td>
<td>Pointer to procedure name</td>
</tr>
<tr>
<td>TCB Obj Name Buf</td>
<td>+1a0</td>
<td>4</td>
<td>D</td>
<td>Pointer to object name buffer</td>
</tr>
<tr>
<td>TCB Tmp Name Buf</td>
<td>+1a4</td>
<td>4</td>
<td>D</td>
<td>aka TCB_Tgt Mod Name Buf</td>
</tr>
<tr>
<td>TCB Src Mod Name Buf</td>
<td>+1a8</td>
<td>4</td>
<td>D</td>
<td>Used by loader</td>
</tr>
</tbody>
</table>
TCB_FaultBuf   +lac  4   D
TCB_ObjNameBufL +lb0  2   W   Length of object name buffer
TCB_TmpNameBufL +lb2  2   W
TCB_SrcModNameBufL +lb4  2   W
TCB_FaultBufL   +lb6  2   W
TCBSecchild     +lb8  4   D   Child Security data 54735

--------------------------------------------
Thread Swappable Data for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRICT kernel

For **TSD** formats for other versions of OS/2 see:

- **TSD** for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel
- **TSD** for OS/2 V2.11 ALLSTRICT kernel
- **TSD** for OS/2 V2.11 RETAIL kernel

**Pointers**

- **TCBpTSD** points to the TSD associated with a TCB
- **CurrTSD** points to the current TSD.

**Locations**

System Arena.

**VM Owner**

- `tst (0xffcd)`

**Format**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSDpad</td>
<td>+0</td>
<td>1000</td>
<td>B</td>
<td>Dummy page to catch faults</td>
</tr>
<tr>
<td>TSDUserStack</td>
<td>+1000</td>
<td>F98</td>
<td>W</td>
<td>Thread's kernel stack</td>
</tr>
<tr>
<td>TSDUserESP</td>
<td>+1f98</td>
<td>4</td>
<td>D</td>
<td>Saved user stack pointer</td>
</tr>
<tr>
<td>TSDUserSS</td>
<td>+1f9c</td>
<td>2</td>
<td>W</td>
<td>Saved user stack segment</td>
</tr>
<tr>
<td>TSDUserSSPad</td>
<td>+1f9e</td>
<td>2</td>
<td>W</td>
<td>Pad word pushed by gate</td>
</tr>
<tr>
<td>TSDKernelESP</td>
<td>+1fa0</td>
<td>4</td>
<td>D</td>
<td>Saved kernel stack pointer</td>
</tr>
<tr>
<td>TSDpTCB</td>
<td>+1fa4</td>
<td>4</td>
<td>D</td>
<td>Link to TCB</td>
</tr>
<tr>
<td>TSDpfnFault</td>
<td>+1fa8</td>
<td>4</td>
<td>D</td>
<td>ptr to local fault handler in effect</td>
</tr>
<tr>
<td>TSDTrapNum</td>
<td>+1fac</td>
<td>4</td>
<td>D</td>
<td>TrapNum from the last fault</td>
</tr>
<tr>
<td>TSDerrcFault</td>
<td>+1fb0</td>
<td>4</td>
<td>D</td>
<td>error code from the last fault</td>
</tr>
<tr>
<td>TSDpljmp</td>
<td>+1fb4</td>
<td>4</td>
<td>D</td>
<td>Buffer saved by TKCatchFault</td>
</tr>
<tr>
<td>TSDselFault</td>
<td>+1fb8</td>
<td>2</td>
<td>W</td>
<td>faulting selector</td>
</tr>
<tr>
<td>TSDCpl2_SSSize</td>
<td>+1fba</td>
<td>2</td>
<td>W</td>
<td>Size of ring 2 stack - atleast thats what the user beleives</td>
</tr>
</tbody>
</table>
Thread Swappable Data for OS/2 Warp V3.0 RETAIL kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSDUserStack</td>
<td>+0</td>
<td>F9C</td>
<td>W</td>
<td>Thread's kernel stack</td>
</tr>
<tr>
<td>TSDUserESP</td>
<td>+f9c</td>
<td>4</td>
<td>D</td>
<td>Saved user stack pointer</td>
</tr>
<tr>
<td>TSDUserSS</td>
<td>+fa0</td>
<td>2</td>
<td>W</td>
<td>Saved user stack segment</td>
</tr>
<tr>
<td>TSDUserSSPad</td>
<td>+fa2</td>
<td>2</td>
<td>W</td>
<td>Pad word pushed by gate</td>
</tr>
<tr>
<td>TSDKernelESP</td>
<td>+fa4</td>
<td>4</td>
<td>D</td>
<td>Saved kernel stack pointer</td>
</tr>
<tr>
<td>TSDpTCB</td>
<td>+fa8</td>
<td>4</td>
<td>D</td>
<td>Link to TCB</td>
</tr>
<tr>
<td>TSDpfnFault</td>
<td>+fac</td>
<td>4</td>
<td>D</td>
<td>ptr to local fault handler in effect</td>
</tr>
<tr>
<td>TSDTrapNum</td>
<td>+fb0</td>
<td>4</td>
<td>D</td>
<td>TrapNum from the last fault</td>
</tr>
<tr>
<td>TSDerrcFault</td>
<td>+fb4</td>
<td>4</td>
<td>D</td>
<td>error code from the last fault</td>
</tr>
<tr>
<td>TSDpljmp</td>
<td>+fb8</td>
<td>4</td>
<td>D</td>
<td>Buffer saved by TKCatchFault</td>
</tr>
<tr>
<td>TSDselFault</td>
<td>+fbc</td>
<td>2</td>
<td>W</td>
<td>faulting selector</td>
</tr>
<tr>
<td>TSDcpl2_SSSize</td>
<td>+fbe</td>
<td>2</td>
<td>W</td>
<td>Size of ring 2 stack - atleast that's what the user believes</td>
</tr>
<tr>
<td>TSDdescLDT</td>
<td>+fc0</td>
<td>8</td>
<td>D</td>
<td>LDT table descriptor</td>
</tr>
<tr>
<td>TSDdescKStackSS</td>
<td>+fc8</td>
<td>8</td>
<td>D</td>
<td>SS descriptor</td>
</tr>
<tr>
<td>TSDdescFPEM</td>
<td>+fd0</td>
<td>8</td>
<td>D</td>
<td>reserved descriptor slot</td>
</tr>
<tr>
<td>TSDdescTIB</td>
<td>+fd8</td>
<td>8</td>
<td>D</td>
<td>FS mapping to TIB</td>
</tr>
<tr>
<td>TSDulExitCode</td>
<td>+fe0</td>
<td>4</td>
<td>D</td>
<td>Proposed Thread Exit code (for dbg)</td>
</tr>
</tbody>
</table>
Thread Swappable Data for OS/2 V2.11 ALLSTRICT kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSDpad</td>
<td>+0</td>
<td>1000</td>
<td>B</td>
<td>Dummy page to catch faults</td>
</tr>
<tr>
<td>TSDUserStack</td>
<td>+1000</td>
<td>F98</td>
<td>W</td>
<td>Thread's kernel stack</td>
</tr>
<tr>
<td>TSDUserESP</td>
<td>+1f98</td>
<td>4</td>
<td>D</td>
<td>Saved stack pointer</td>
</tr>
<tr>
<td>TSDUserSS</td>
<td>+1f9c</td>
<td>2</td>
<td>W</td>
<td>Saved stack segment</td>
</tr>
<tr>
<td>TSDUserSSPad</td>
<td>+1f9e</td>
<td>2</td>
<td>W</td>
<td>Pad word pushed by gate</td>
</tr>
<tr>
<td>TSDKernelESP</td>
<td>+1fa0</td>
<td>4</td>
<td>D</td>
<td>Saved kernel stack pointer</td>
</tr>
<tr>
<td>TSDpTCB</td>
<td>+1fa4</td>
<td>4</td>
<td>D</td>
<td>Link to TCB</td>
</tr>
<tr>
<td>TSDfnFault</td>
<td>+1fa8</td>
<td>4</td>
<td>D</td>
<td>Pointer to local fault handler in effect</td>
</tr>
<tr>
<td>TSTDtrapNum</td>
<td>+1fac</td>
<td>4</td>
<td>D</td>
<td>TrapNum from the last fault</td>
</tr>
<tr>
<td>TSDerrcFault</td>
<td>+1fb0</td>
<td>4</td>
<td>D</td>
<td>Error code from the last fault</td>
</tr>
<tr>
<td>TSDpljmp</td>
<td>+1fb4</td>
<td>4</td>
<td>D</td>
<td>Buffer saved by TKCatchFault</td>
</tr>
<tr>
<td>TSDselFault</td>
<td>+1fb8</td>
<td>2</td>
<td>W</td>
<td>Faulting selector</td>
</tr>
<tr>
<td>TSDLastL2_SSSize</td>
<td>+1fba</td>
<td>2</td>
<td>W</td>
<td>Size of ring 2 stack - at least what the user believes</td>
</tr>
<tr>
<td>TSDdescLDT</td>
<td>+1fbc</td>
<td>8</td>
<td>D</td>
<td>LDT table descriptor</td>
</tr>
<tr>
<td>TSDdescKStackSS</td>
<td>+1fc4</td>
<td>8</td>
<td>D</td>
<td>SS descriptor</td>
</tr>
<tr>
<td>TSDdescFPEM</td>
<td>+1fcc</td>
<td>8</td>
<td>D</td>
<td>Reserved descriptor slot</td>
</tr>
<tr>
<td>TSDdescTIB</td>
<td>+1fd4</td>
<td>8</td>
<td>D</td>
<td>FS mapping to TIB</td>
</tr>
<tr>
<td>TSDuExitCode</td>
<td>+1fdc</td>
<td>4</td>
<td>D</td>
<td>Proposed Thread Exit code (for dbg)</td>
</tr>
<tr>
<td>TSDerridFault</td>
<td>+1fe0</td>
<td>4</td>
<td>D</td>
<td>Error id from page fault</td>
</tr>
<tr>
<td>TSDPFErr</td>
<td>+1fe4</td>
<td>4</td>
<td>D</td>
<td>Actual error from PGPageFault</td>
</tr>
<tr>
<td>TSDlDbgRangeStart</td>
<td>+1fe8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSDlDbgRangeEnd</td>
<td>+1fec</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSDlDbgLastAddr</td>
<td>+1ff0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSDpPCB</td>
<td>+1ff4</td>
<td>4</td>
<td>D</td>
<td>Pointer to Profile Control Block</td>
</tr>
<tr>
<td>TSDpDLLTerm</td>
<td>+1ff8</td>
<td>4</td>
<td>D</td>
<td>Pointer to data buffer</td>
</tr>
</tbody>
</table>
Thread Swappable Data for OS/2 V2.11 RETAIL kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSDUserStack</td>
<td>+0</td>
<td>F9C</td>
<td>W</td>
<td>Thread's kernel stack</td>
</tr>
<tr>
<td>TSDUserESP</td>
<td>+f9c</td>
<td>4</td>
<td>D</td>
<td>Saved user stack pointer</td>
</tr>
<tr>
<td>TSDUserSS</td>
<td>+fa0</td>
<td>2</td>
<td>W</td>
<td>Saved user stack segment</td>
</tr>
<tr>
<td>TSDUserSSPad</td>
<td>+fa2</td>
<td>2</td>
<td>W</td>
<td>Pad word pushed by gate</td>
</tr>
<tr>
<td>TSDKernelESP</td>
<td>+fa4</td>
<td>4</td>
<td>D</td>
<td>Saved kernel stack pointer.</td>
</tr>
<tr>
<td>TSDpTCB</td>
<td>+fa8</td>
<td>4</td>
<td>D</td>
<td>Link to TCB</td>
</tr>
<tr>
<td>TSDpfnFault</td>
<td>+fac</td>
<td>4</td>
<td>D</td>
<td>ptr to local fault handler in effect</td>
</tr>
<tr>
<td>TSDTrapNum</td>
<td>+fb0</td>
<td>4</td>
<td>D</td>
<td>TrapNum from the last fault</td>
</tr>
<tr>
<td>TSDerrcFault</td>
<td>+fb4</td>
<td>4</td>
<td>D</td>
<td>error code from the last fault</td>
</tr>
<tr>
<td>TSDpljmp</td>
<td>+fb8</td>
<td>4</td>
<td>D</td>
<td>Buffer saved by TKCatchFault</td>
</tr>
<tr>
<td>TSDselFault</td>
<td>+fbc</td>
<td>2</td>
<td>W</td>
<td>faulting selector</td>
</tr>
<tr>
<td>TSDCpl2_SSSize</td>
<td>+fbe</td>
<td>2</td>
<td>W</td>
<td>Size of ring 2 stack - atleast thats what the user believes</td>
</tr>
<tr>
<td>TSDdescLDT</td>
<td>+fc0</td>
<td>8</td>
<td>D</td>
<td>LDT table descriptor</td>
</tr>
<tr>
<td>TSDescKStackSS</td>
<td>+fc8</td>
<td>8</td>
<td>D</td>
<td>SS descriptor</td>
</tr>
<tr>
<td>TSDdescFPEM</td>
<td>+fd0</td>
<td>8</td>
<td>D</td>
<td>reserved descriptor slot</td>
</tr>
<tr>
<td>TSDdescTIB</td>
<td>+fd8</td>
<td>8</td>
<td>D</td>
<td>FS mapping to TIB</td>
</tr>
<tr>
<td>TSDu1ExitCode</td>
<td>+fe0</td>
<td>4</td>
<td>D</td>
<td>Proposed Thread Exit code (for dbg)</td>
</tr>
<tr>
<td>TSDerridFault</td>
<td>+fe4</td>
<td>4</td>
<td>D</td>
<td>error id from page fault</td>
</tr>
<tr>
<td>TSDPFErr</td>
<td>+fe8</td>
<td>4</td>
<td>D</td>
<td>actual error from PGPagefault</td>
</tr>
<tr>
<td>TSD1DbgRangeStart</td>
<td>+fec</td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSD1DbgRangeEnd</td>
<td>+ff0</td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSD1DbgLastAddr</td>
<td>+ff4</td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>TSDpPCB</td>
<td>+ff8</td>
<td>4</td>
<td>D</td>
<td>Pointer to Profile Control Block</td>
</tr>
<tr>
<td>TSDpDLLTerm</td>
<td>+ffc</td>
<td>4</td>
<td>D</td>
<td>Pointer to data buffer</td>
</tr>
</tbody>
</table>

Local Exception Handler Long-Jump Buffer
Points

TSDpljmp points to current reistered ljmp buffer.

ljmp_pljmp points to the next nested ljmp buffer.

Locations

System Arena. Usually allocated as local data on the Ring 0 stack.

VM Owner

tsd (0xffcd)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ljmp_lEBX</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>EBX restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_lESI</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>ESI restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_lEDI</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>EDI restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_lEBP</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>EBP restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_lESP</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>ESP restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_lEIP</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>EIP restored when local exception handler returns control</td>
</tr>
<tr>
<td>ljmp_pfnFault</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Address of previous fault handler</td>
</tr>
<tr>
<td>ljmp_pljmp</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Address of previous long jump buffer</td>
</tr>
</tbody>
</table>

--------------------------------------------

Per-Task Data Area for OS/2 Warp V4.0 ALLSTRICT kernel

For PTDA formats for other versions of OS/2 see:

PTDA for OS/2 Warp V4.0 RETAIL kernel
PTDA for OS/2 Warp V3.0 ALLSTRICT kernel
PTDA for OS/2 Warp V3.0 RETAIL kernel
PTDA for OS/2 V2.11 ALLSTRICT kernel
PTDA for OS/2 V2.11 RETAIL kernel

Pointers

TCBpPTDA points to the PTDA associated with a TCB

CurTSFornts points to the current TSD.

pPTDASelf points to the current PTDA.

Locations

System Arena.
## Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDAParent</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Parent PTDA</td>
</tr>
<tr>
<td>pPTDAself</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>This PTDA</td>
</tr>
<tr>
<td>pPTDAFirstChild</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Head of child chain PTDA</td>
</tr>
<tr>
<td>pPTDAExecChild</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>New Child PTDA (Child being exec'ed)</td>
</tr>
<tr>
<td>pPTDANextSibling</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Next sibling's PTDA</td>
</tr>
<tr>
<td>pPTDAPrevSibling</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Previous sibling's PTDA</td>
</tr>
<tr>
<td>ptda_pszproc</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to the EXE file this process is executing. Used by PerfView</td>
</tr>
<tr>
<td>ptda_pTCBHole</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>some TCB before first Tid 'hole'</td>
</tr>
<tr>
<td>ptda_pTCBHead</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Head of list of active TCBs owned by this process</td>
</tr>
<tr>
<td>ptda_cTCB</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Number of TCBs in use</td>
</tr>
<tr>
<td>ptda_ctib</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td>Count of TIBs allocated</td>
</tr>
<tr>
<td>ptda_avatib</td>
<td>+28</td>
<td>10</td>
<td>D</td>
<td>Pointers to TIB arrays</td>
</tr>
<tr>
<td>ptda_pdcb</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_flDbg</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_ah</td>
<td>+40</td>
<td>40</td>
<td>S</td>
<td>Private arena header</td>
</tr>
<tr>
<td>ptda_pgdata</td>
<td>+80</td>
<td>26</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_environ</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td>handle to process's envt seg</td>
</tr>
<tr>
<td>ptda_pBeginLIBPATH</td>
<td>+a8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pEndLIBPATH</td>
<td>+ac</td>
<td>4</td>
<td>D</td>
<td>D75220- support dynamic libpath</td>
</tr>
<tr>
<td>ptda_pgpc</td>
<td>+b0</td>
<td>240</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pPVDBPrc</td>
<td>+2f0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pSGSList</td>
<td>+2f4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pexllist</td>
<td>+2f8</td>
<td>4</td>
<td>D</td>
<td>Flat pointer to exit list data</td>
</tr>
<tr>
<td>ptda_cd1lterm</td>
<td>+2fc</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>CDS_Handle</td>
<td>+300</td>
<td>34</td>
<td>W</td>
<td>array of current directory handles</td>
</tr>
<tr>
<td>OEMPtr</td>
<td>+334</td>
<td>2</td>
<td>W</td>
<td>Offset to OEM-Added fields</td>
</tr>
<tr>
<td>VerFlg</td>
<td>+336</td>
<td>1</td>
<td>B</td>
<td>Initialize with verify off</td>
</tr>
<tr>
<td>LCurDrv</td>
<td>+337</td>
<td>1</td>
<td>B</td>
<td>Logical current drive - Default A:</td>
</tr>
<tr>
<td>PCurDrv</td>
<td>+338</td>
<td>1</td>
<td>B</td>
<td>physical drive after assign mapping</td>
</tr>
<tr>
<td>LIS_Fgnd</td>
<td>+339</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FgndOnly</td>
<td>+33a</td>
<td>1</td>
<td>B</td>
<td>foreground only flag</td>
</tr>
<tr>
<td>ptda_pad1</td>
<td>+33b</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ptda_pTCBCritSec</td>
<td>+33c</td>
<td>4</td>
<td>D</td>
<td>TCB that did enter CritSec</td>
</tr>
<tr>
<td>ptda_pTCBPriQCritSec</td>
<td>+340</td>
<td>4</td>
<td>D</td>
<td>TCBs awaiting CritSec wakeup</td>
</tr>
</tbody>
</table>
ptda_cCritSec +344 2 W Critical Section Count
CurrentPDB   +346 2 W Currently active PDB (V86 segment)
seltss      +348 2 W
ProcFlag    +34a 2 W if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.

ptda_ForceActions +34c 4 D pending action bits
ptda_ulExitCode  +350 4 D Exit code of last task
ptda_ulExitType  +354 4 D Type of exit
ptda_ulExitTID   +358 4 D Exit Thread ID (32-bit exceptions)
ThisCDS        +35c 4 D Address of current CDS *REDIR* 3.10
ptda_pCDS       +360 2 W SS relative pointer to a curdir struct
CDSsize        +362 2 W Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib        +364 2 W Storage for search attrs *REDIR* 3.10
sPCB           +366 2 W Selector of Profile Control Block
ptda_pPCB      +368 4 D Pointer to Profile Control Block
JFN_Max        +36c 2 W highest JFN used so far
NextSrchH      +36e 2 W Next value to use for search handle First value used will be 2.
SrchRmp        +370 4 D Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_first  +374 2 W
FNotifyLocal_Count +376 2 W
Sig_ignf        +378 2 W bit vector of ignored signals
Sig_hndf        +37a 2 W bit vector of handled signals
Sig_errf        +37c 2 W bit vector of error generating signals
Sig_attempted   +37e 2 W bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg         +380 10 W byte vector of signal arguments
Sig_termtid     +390 2 W 'Terminator' TID for ATERM.
HoldSigCnt      +392 2 W DOSHOLDIGNAL counter
SigFocusCnt     +394 2 W PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table       +396 28 W default handle table
JFN_Flags       +3be 2 B default JFN flags table
ptda_rasflag    +3d2 14 B default JFN flags table
SysSemPTDATbl   +3d4 100 S
SavedHardErr    +4d4 4 D
ptda_ptdasem    +4d8 6 S PTDA semaphore that is, inter-thread
ptda_DLLsem     +4e4 6 S b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt      +4f0 6 W current IDT limit/base
<table>
<thead>
<tr>
<th>Variable</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CsId</td>
<td>+4f6</td>
<td>2</td>
<td>W</td>
<td>Command Subtree ID</td>
</tr>
<tr>
<td>Behav_bit</td>
<td>+4f8</td>
<td>2</td>
<td>W</td>
<td>program behavior bits</td>
</tr>
<tr>
<td>MSW</td>
<td>+4fa</td>
<td>2</td>
<td>W</td>
<td>CPU matching status word</td>
</tr>
<tr>
<td>ptda_rsrsclst</td>
<td>+4fc</td>
<td>4</td>
<td>D</td>
<td>far pointer to local resource list</td>
</tr>
<tr>
<td>ptda_pldrdldHead</td>
<td>+500</td>
<td>4</td>
<td>D</td>
<td>loader demand load data list</td>
</tr>
<tr>
<td>pPrSemTbl</td>
<td>+504</td>
<td>4</td>
<td>D</td>
<td>(void * =&gt; PSEM) pointer to private semaphore table</td>
</tr>
<tr>
<td>ulPrTblSize</td>
<td>+508</td>
<td>4</td>
<td>D</td>
<td>size of pPrSemTbl in dwords</td>
</tr>
<tr>
<td>ulPrTotUsed</td>
<td>+50c</td>
<td>4</td>
<td>D</td>
<td>number of entries in pPrSemTbl</td>
</tr>
<tr>
<td>ulPrNextFree</td>
<td>+510</td>
<td>4</td>
<td>D</td>
<td>next free slot in pPrSemTbl</td>
</tr>
<tr>
<td>hksPrTbl</td>
<td>+514</td>
<td>4</td>
<td>D</td>
<td>kernel semaphore handle for private semaphore table</td>
</tr>
<tr>
<td>pShSemBmp</td>
<td>+518</td>
<td>4</td>
<td>D</td>
<td>pointer to private bitmap for the shared semaphore table</td>
</tr>
<tr>
<td>ulShBmpSize</td>
<td>+51c</td>
<td>4</td>
<td>D</td>
<td>size of pShSemBmp in bits</td>
</tr>
<tr>
<td>hksShBmp</td>
<td>+520</td>
<td>4</td>
<td>D</td>
<td>kernel semaphore handle for private semaphore table</td>
</tr>
<tr>
<td>ulMtxOwned</td>
<td>+524</td>
<td>4</td>
<td>D</td>
<td>number of mutex owned by this process in the two sem tables</td>
</tr>
<tr>
<td>ptda_TLMA</td>
<td>+528</td>
<td>4</td>
<td>D</td>
<td>in use flag and dword copy count</td>
</tr>
<tr>
<td>ptda_TLMABM</td>
<td>+52c</td>
<td>4</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>ptda_TLMASizemap</td>
<td>+530</td>
<td>20</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>Cons_Loc</td>
<td>+550</td>
<td>A</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SysCallSfcn</td>
<td>+55a</td>
<td>1</td>
<td>B</td>
<td>Value of AL on system entry</td>
</tr>
<tr>
<td>SysCall</td>
<td>+55b</td>
<td>1</td>
<td>B</td>
<td>Last system call processed</td>
</tr>
<tr>
<td>KBD_Mode</td>
<td>+55c</td>
<td>1</td>
<td>B</td>
<td>Keyboard input mode</td>
</tr>
<tr>
<td>ptda_NewFiles</td>
<td>+55d</td>
<td>1</td>
<td>B</td>
<td>If bit one is set, process supports // 54400 new files (long names)</td>
</tr>
<tr>
<td>AutoFail</td>
<td>+55e</td>
<td>1</td>
<td>B</td>
<td>Non-zero if I 24 FAILED magically</td>
</tr>
<tr>
<td>CP_Flags</td>
<td>+55f</td>
<td>1</td>
<td>B</td>
<td>Default is no codepage in system.</td>
</tr>
<tr>
<td>Sig_vec</td>
<td>+560</td>
<td>20</td>
<td>D</td>
<td>signal handlers</td>
</tr>
<tr>
<td>Exc_vec</td>
<td>+580</td>
<td>1C</td>
<td>D</td>
<td>obsolete exception vectors</td>
</tr>
<tr>
<td>ptda_timerhead</td>
<td>+59c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_extsig</td>
<td>+5a0</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ptda_pad6</td>
<td>+5a1</td>
<td>3</td>
<td>B</td>
<td>alignment</td>
</tr>
<tr>
<td>pPvwDataBlk</td>
<td>+5a4</td>
<td>4</td>
<td>D</td>
<td>Used by perfview</td>
</tr>
<tr>
<td>ptda_lanman_sec</td>
<td>+5a8</td>
<td>4</td>
<td>D</td>
<td>Used by LANMAN &amp; HPFS for security.</td>
</tr>
<tr>
<td>SigFTerm</td>
<td>+5ac</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_ppgdata</td>
<td>+5ae</td>
<td>2</td>
<td>W</td>
<td>offset ptda_pgdta</td>
</tr>
<tr>
<td>ptda_child</td>
<td>+5b0</td>
<td>2</td>
<td>W</td>
<td>New child PTDA handle (Child being Exec'ed)</td>
</tr>
<tr>
<td>ptda_childdallas</td>
<td>+5b2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_handle</td>
<td>+5b4</td>
<td>2</td>
<td>W</td>
<td>handle to this segment</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ptda_module</td>
<td>+5b6</td>
<td>2</td>
<td>W</td>
<td>program module handle for process</td>
</tr>
<tr>
<td>ptda_lthandle</td>
<td>+5b8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_ldtpgmap</td>
<td>+5ba</td>
<td>2</td>
<td>W</td>
<td>Bitmap of valid LDT pages</td>
</tr>
<tr>
<td>ptda_ldtaddr</td>
<td>+5bc</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>CP_CaseMapTbl</td>
<td>+5c0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>codepage_tag</td>
<td>+5c4</td>
<td>2</td>
<td>W</td>
<td>the current code page</td>
</tr>
<tr>
<td>JFN_Length</td>
<td>+5c6</td>
<td>2</td>
<td>W</td>
<td>Size of JFN table in bytes</td>
</tr>
<tr>
<td>JFN_pTable</td>
<td>+5c8</td>
<td>4</td>
<td>D</td>
<td>PM pointer to JFN table</td>
</tr>
<tr>
<td>JFN_Flg_Ptr</td>
<td>+5cc</td>
<td>4</td>
<td>D</td>
<td>pointer to JFN flags</td>
</tr>
<tr>
<td>ptda_pad</td>
<td>+5d0</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ExtErr_Locus</td>
<td>+5d1</td>
<td>1</td>
<td>B</td>
<td>Extended Error Locus <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr</td>
<td>+5d2</td>
<td>2</td>
<td>W</td>
<td>Extended Error code <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr_Action</td>
<td>+5d4</td>
<td>1</td>
<td>B</td>
<td>Extended Error Action <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr_Class</td>
<td>+5d5</td>
<td>1</td>
<td>B</td>
<td>Extended Error Class <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ptda_infoseg</td>
<td>+5d6</td>
<td>24</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_vme</td>
<td>+5fa</td>
<td>1</td>
<td>B</td>
<td>VME Flag</td>
</tr>
<tr>
<td>ptda_pad3</td>
<td>+5fb</td>
<td>1</td>
<td>B</td>
<td>alignment</td>
</tr>
<tr>
<td>CurrTCB</td>
<td>+5fc</td>
<td>2</td>
<td>W</td>
<td>pointer to current TCB</td>
</tr>
<tr>
<td>CurrTSD</td>
<td>+5fe</td>
<td>2</td>
<td>W</td>
<td>pointer to current TSD</td>
</tr>
<tr>
<td>ThisPTDA</td>
<td>+600</td>
<td>2</td>
<td>W</td>
<td>Selector for this ptda</td>
</tr>
<tr>
<td>ptda_NPX_em_cs</td>
<td>+602</td>
<td>2</td>
<td>W</td>
<td>b726833 NPX emulator CS b726833</td>
</tr>
<tr>
<td>ptda_NPX_em_eip</td>
<td>+604</td>
<td>4</td>
<td>D</td>
<td>b726833 NPX emulator EIP b726833</td>
</tr>
<tr>
<td>ptda_pad4</td>
<td>+608</td>
<td>2</td>
<td>W</td>
<td>alignment b726833</td>
</tr>
<tr>
<td>ptda_signature</td>
<td>+60a</td>
<td>2</td>
<td>B</td>
<td>must contain &quot;TD&quot;</td>
</tr>
</tbody>
</table>

**ptda_ForcedActions** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK_FF_BUF</td>
<td>0x00000001</td>
<td>Buffer must be released</td>
</tr>
<tr>
<td>TK_FF_EXIT</td>
<td>0x00000002</td>
<td>Call TKExit (old FF_DES)</td>
</tr>
<tr>
<td>TK_FF_CRITSEC</td>
<td>0x00000004</td>
<td>Enter Per-task critical section</td>
</tr>
<tr>
<td>TK_FF_ICE</td>
<td>0x00000008</td>
<td>Freeze thread</td>
</tr>
<tr>
<td>TK_FF_NPX</td>
<td>0x00000010</td>
<td>NPX Error</td>
</tr>
<tr>
<td>TK_FF_TIB</td>
<td>0x00000020</td>
<td>Update the TIB</td>
</tr>
<tr>
<td>TK_FF_TRC</td>
<td>0x00000040</td>
<td>Enter Debug</td>
</tr>
<tr>
<td>TK_FF_SIG</td>
<td>0x00000080</td>
<td>Signal pending</td>
</tr>
<tr>
<td>TK_FF_CTXH</td>
<td>0x00000100</td>
<td>Pending local context hooks</td>
</tr>
<tr>
<td>TK_FF_STIH</td>
<td>0x00000200</td>
<td>Execute STI hooks</td>
</tr>
<tr>
<td>TK_FF_VDMBP</td>
<td>0x00000400</td>
<td>Execute VDM BP hooks</td>
</tr>
</tbody>
</table>
Per-Task Data Area for OS/2 Warp V4.0 RETAIL kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDAParent</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Parent PTDA</td>
</tr>
<tr>
<td>pPTDASelf</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>This PTDA</td>
</tr>
<tr>
<td>pPTDAFirstChild</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Head of child chain PTDA</td>
</tr>
<tr>
<td>pPTDAExecChild</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>New Child PTDA (Child being exec'ed)</td>
</tr>
<tr>
<td>pPTDANextSibling</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Next sibling's PTDA</td>
</tr>
<tr>
<td>pPTDAPrevSibling</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Previous sibling's PTDA</td>
</tr>
<tr>
<td>ptda_pszproc</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to the EXE file this process is executing. Used by PerfView</td>
</tr>
<tr>
<td>ptda_pTCBHole</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>some TCB before first Tid 'hole'</td>
</tr>
<tr>
<td>ptda_pTCBHead</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Head of list of active TCBs owned by this process</td>
</tr>
<tr>
<td>ptda_cTCB</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Number of TCBs in use</td>
</tr>
<tr>
<td>ptda_ctib</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td>Count of TIBs allocated</td>
</tr>
<tr>
<td>ptda_avatib</td>
<td>+28</td>
<td>10</td>
<td>D</td>
<td>Pointers to TIB arrays</td>
</tr>
<tr>
<td>ptda_pdcb</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_flDbg</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_ah</td>
<td>+40</td>
<td>40</td>
<td>S</td>
<td>Private arena header</td>
</tr>
<tr>
<td>ptda_pgdata</td>
<td>+80</td>
<td>26</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_environ</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td>handle to process's envt seg</td>
</tr>
<tr>
<td>ptda_pBeginLIBPATH</td>
<td>+a8</td>
<td>4</td>
<td>D</td>
<td>D75220- support dynamic libpath</td>
</tr>
<tr>
<td>ptda_pEndLIBPATH</td>
<td>+ac</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pgppc</td>
<td>+b0</td>
<td>240</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pPVDBPrc</td>
<td>+2f0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pSQLList</td>
<td>+2f4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pexllist</td>
<td>+2f8</td>
<td>4</td>
<td>D</td>
<td>Flat pointer to exit list data</td>
</tr>
<tr>
<td>ptda_cdltterm</td>
<td>+2fc</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>CDS_Handle</td>
<td>+300</td>
<td>34</td>
<td>W</td>
<td>array of current directory handles</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OEMPtr</td>
<td>+334</td>
<td>2</td>
<td>W</td>
<td>Offset to OEM-Added fields</td>
</tr>
<tr>
<td>VerFlg</td>
<td>+336</td>
<td>1</td>
<td>B</td>
<td>Initialize with verify off</td>
</tr>
<tr>
<td>LCurDrv</td>
<td>+337</td>
<td>1</td>
<td>B</td>
<td>Logical current drive - Default A:</td>
</tr>
<tr>
<td>PCurDrv</td>
<td>+338</td>
<td>1</td>
<td>B</td>
<td>physical drive after assign mapping</td>
</tr>
<tr>
<td>LIS_Fgnd</td>
<td>+339</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FgndOnly</td>
<td>+33a</td>
<td>1</td>
<td>B</td>
<td>foreground only flag</td>
</tr>
<tr>
<td>ptda_pad1</td>
<td>+33b</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ptda_pTCBCritSec</td>
<td>+33c</td>
<td>4</td>
<td>D</td>
<td>TCB that did enter CritSec</td>
</tr>
<tr>
<td>ptda_pTCBPriQCritSec</td>
<td>+340</td>
<td>4</td>
<td>D</td>
<td>TCBs awaiting CritSec wakeup</td>
</tr>
<tr>
<td>ptda_cCritSec</td>
<td>+344</td>
<td>2</td>
<td>W</td>
<td>Critical Section Count</td>
</tr>
<tr>
<td>CurrentPDB</td>
<td>+346</td>
<td>2</td>
<td>W</td>
<td>Currently active PDB (V86 segment)</td>
</tr>
<tr>
<td>selts</td>
<td>+348</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ProcFlag</td>
<td>+34a</td>
<td>2</td>
<td>W</td>
<td>if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.</td>
</tr>
<tr>
<td>ptda_ForceActions</td>
<td>+34c</td>
<td>4</td>
<td>D</td>
<td>pending action bits</td>
</tr>
<tr>
<td>ptda_u1ExitCode</td>
<td>+350</td>
<td>4</td>
<td>D</td>
<td>Exit code of last task</td>
</tr>
<tr>
<td>ptda_u1ExitType</td>
<td>+354</td>
<td>4</td>
<td>D</td>
<td>Type of exit</td>
</tr>
<tr>
<td>ptda_u1ExitTID</td>
<td>+358</td>
<td>4</td>
<td>D</td>
<td>Exit Thread ID (32-bit exceptions)</td>
</tr>
<tr>
<td>ThisCDS</td>
<td>+35c</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ptda_pCDS</td>
<td>+360</td>
<td>2</td>
<td>W</td>
<td>SS relative pointer to a curdir struct</td>
</tr>
<tr>
<td>CDSsize</td>
<td>+362</td>
<td>2</td>
<td>W</td>
<td>Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg</td>
</tr>
<tr>
<td>Sattrib</td>
<td>+364</td>
<td>2</td>
<td>W</td>
<td>Storage for search attrs <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>sPCB</td>
<td>+366</td>
<td>2</td>
<td>W</td>
<td>Selector of Profile Control Block</td>
</tr>
<tr>
<td>ptda_pPCB</td>
<td>+368</td>
<td>4</td>
<td>D</td>
<td>Pointer to Profile Control Block</td>
</tr>
<tr>
<td>JFN_Max</td>
<td>+36c</td>
<td>2</td>
<td>W</td>
<td>highest JFN used so far</td>
</tr>
<tr>
<td>NextSrchH</td>
<td>+36e</td>
<td>2</td>
<td>W</td>
<td>Next value to use for search handle First value used will be 2.</td>
</tr>
<tr>
<td>SrchRmp</td>
<td>+370</td>
<td>4</td>
<td>D</td>
<td>Handle &amp; Selector for RMP segment we keep search handles in.</td>
</tr>
<tr>
<td>FNotifyLocal_First</td>
<td>+374</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>FNotifyLocal_Count</td>
<td>+376</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>Sig_ignf</td>
<td>+378</td>
<td>2</td>
<td>W</td>
<td>bit vector of ignored signals</td>
</tr>
<tr>
<td>Sig_hndf</td>
<td>+37a</td>
<td>2</td>
<td>W</td>
<td>bit vector of handled signals</td>
</tr>
<tr>
<td>Sig_errf</td>
<td>+37c</td>
<td>2</td>
<td>W</td>
<td>bit vector of error generating signals</td>
</tr>
<tr>
<td>Sig_attempted</td>
<td>+37e</td>
<td>2</td>
<td>W</td>
<td>bit vector of signals we've tried to handle with 32-bit exceptions</td>
</tr>
<tr>
<td>Sig_arg</td>
<td>+380</td>
<td>10</td>
<td>W</td>
<td>byte vector of signal arguments</td>
</tr>
<tr>
<td>Sig_termtid</td>
<td>+390</td>
<td>2</td>
<td>W</td>
<td>'Terminator' TID for ATERM.</td>
</tr>
<tr>
<td>HoldSigCnt</td>
<td>+392</td>
<td>2</td>
<td>W</td>
<td>DOSHOLD SIGNAL counter</td>
</tr>
<tr>
<td>SigFocusCnt</td>
<td>+394</td>
<td>2</td>
<td>W</td>
<td>PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS</td>
</tr>
</tbody>
</table>
JFN_Table            +396   28     W    default handle table
JFN_Flags            +3be   14     B    default JFN flags table
ptda_rasflag         +3d2   2      W    RAS trace indicator
SysSemPTDATbl        +3d4   100    S
SavedHardErr         +4d4   4      D
ptda_ptdasem         +4d8   8      S    PTDA semaphore that is, inter-thread
ptda_DLMsem          +4e0   8      S    b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt            +4e8   6     W    current IDT limit/base
Csid                 +4ee   2      W    Command Subtree ID
Behav_bit            +4f0   2      W    program behavior bits
MSW                  +4f2   2      W    CPU matching status word
ptda_rsrclist        +4f4   4      D    far pointer to local resource list
ptda_pldrddldHead     +4f8   4      D    loader demand load data list
pPrSemTbl            +4fc   4      D    (void * => PSEM) pointer to private semaphore table
ulPrTblSize          +500   4      D    size of pPrSemTbl in dwords
ulPrTotUsed          +504   4      D    number of entries in pPrSemTbl
ulPrNextFree         +508   4      D    next free slot in pPrSemTbl
hksPrTbl             +50c   4      D    kernel semaphore handle for private semaphore table
pShSemBmp            +510   4      D    pointer to private bitmap for the shared semaphore table
ulShBmpSize          +514   4      D    size of pShSemBmp in bits
hksShBmp             +518   4      D    kernel semaphore handle for private semaphore table
ulMtxOwned           +51c   4      D    number of mutex owned by this process in the two sem tables
ptda_TLMA            +520   4      D    in use flag and dword copy count
ptda_TLMABM          +524   4      B    thread local memory
ptda_TLMASizeMap     +528   20     B    thread local memory
Cons_Loc             +548   A      S
SysCallSfcn          +552   1      B    Value of AL on system entry
SysCall              +553   1      B    Last system call processed
KBD_Mode             +554   1      B    Keyboard input mode
ptda_NewFiles        +555   1      B    If bit one is set, process supports // 54400 new files (long names)
AutoFail             +556   1      B    Non-zero if I 24 FAILED magically
CP_Flgs              +557   1      B    Default is no codepage in system.
Sig_vec              +558   20     D    signal handlers
Exc_vec              +578   1C     D    OSOLETE exception vectors
ptda_timerhead       +594   4      D
Per-Task Data Area for OS/2 Warp V3.0 ALLSTRICT kernel
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDAParent</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Parent PTDA</td>
</tr>
<tr>
<td>pPTDA себе</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>This PTDA</td>
</tr>
<tr>
<td>pPTDAFirstChild</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Head of child chain PTDA</td>
</tr>
<tr>
<td>pPTDAExecChild</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>New Child PTDA (Child being exec'ed)</td>
</tr>
<tr>
<td>pPTDANextSibling</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Next sibling's PTDA</td>
</tr>
<tr>
<td>pPTDAPrevSibling</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Previous sibling's PTDA</td>
</tr>
<tr>
<td>ptda_pszproc</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to the EXE file this process is executing. Used by PerfView</td>
</tr>
<tr>
<td>ptda_pTCBHole</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>some TCB before first Tid 'hole'</td>
</tr>
<tr>
<td>ptda_pTCBHead</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Head of list of active TCBs owned by this process</td>
</tr>
<tr>
<td>ptda_cTCB</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Number of TCBs in use</td>
</tr>
<tr>
<td>ptda_ctlib</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td>Count of TIBs allocated</td>
</tr>
<tr>
<td>ptda_avatib</td>
<td>+28</td>
<td>10</td>
<td>D</td>
<td>Pointers to TIB arrays</td>
</tr>
<tr>
<td>ptda_pdcb</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_fldbg</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_ah</td>
<td>+40</td>
<td>40</td>
<td>S</td>
<td>Private arena header</td>
</tr>
<tr>
<td>ptda_pgdata</td>
<td>+80</td>
<td>26</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_environ</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td>handle to process's envt seg</td>
</tr>
<tr>
<td>ptda_pBeginLIBPATH</td>
<td>+a8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pEndLIBPATH</td>
<td>+ac</td>
<td>4</td>
<td>D</td>
<td>D75220- support dynamic libpath</td>
</tr>
<tr>
<td>ptda_ppgc</td>
<td>+b0</td>
<td>1E0</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pPVDBPrc</td>
<td>+290</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pSGSLList</td>
<td>+294</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pexllist</td>
<td>+298</td>
<td>4</td>
<td>D</td>
<td>Flat pointer to exit list data</td>
</tr>
<tr>
<td>ptda_cdllterm</td>
<td>+29c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>WFP_Start</td>
<td>+2a0</td>
<td>2</td>
<td>W</td>
<td>TASKAREA offset for working string <em>REDIR</em></td>
</tr>
<tr>
<td><em>REDIR</em> Ren_WFP</td>
<td>+2a2</td>
<td>2</td>
<td>W</td>
<td>WFB pointer for rename destination <em>REDIR</em></td>
</tr>
<tr>
<td>WFP_Path_End</td>
<td>+2a4</td>
<td>2</td>
<td>W</td>
<td>End of Path component of string.</td>
</tr>
<tr>
<td>Curr_Dir_End</td>
<td>+2a6</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>CDS_Handle</td>
<td>+2a8</td>
<td>34</td>
<td>W</td>
<td><em>REDIR</em></td>
</tr>
<tr>
<td>OEMPtr</td>
<td>+2dc</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>LIS_Fgnd</td>
<td>+2de</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FgndOnly</td>
<td>+2df</td>
<td>1</td>
<td>B</td>
<td>foreground only flag</td>
</tr>
<tr>
<td>ptda_pTCBCritSec</td>
<td>+2e0</td>
<td>4</td>
<td>D</td>
<td>TCB that did enter CritSec</td>
</tr>
<tr>
<td>ptda_pTCBPriQCritSec</td>
<td>+2e4</td>
<td>4</td>
<td>D</td>
<td>TCBs awaiting CritSec wakeup</td>
</tr>
<tr>
<td>ptda_cCritSec</td>
<td>+2e8</td>
<td>2</td>
<td>W</td>
<td>Critical Section Count</td>
</tr>
<tr>
<td>CurrentPDB</td>
<td>+2ea</td>
<td>2</td>
<td>W</td>
<td>Currently active PDB (V86 segment)</td>
</tr>
<tr>
<td>Field</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DTAddr</td>
<td>+2ec</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address <em>REDIR</em></td>
</tr>
<tr>
<td>selts</td>
<td>+2f0</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>VolID</td>
<td>+2f2</td>
<td>1</td>
<td>B</td>
<td>0 if vol ID found in dir search</td>
</tr>
<tr>
<td>NoSetDir</td>
<td>+2f3</td>
<td>1</td>
<td>B</td>
<td>If TRUE, do not set directory</td>
</tr>
<tr>
<td>SpaceFlag</td>
<td>+2f4</td>
<td>1</td>
<td>B</td>
<td>Embedded spaces allowed in FCB</td>
</tr>
<tr>
<td>VerFlg</td>
<td>+2f5</td>
<td>1</td>
<td>B</td>
<td>Initialize with verify off</td>
</tr>
<tr>
<td>LCurDrv</td>
<td>+2f6</td>
<td>1</td>
<td>B</td>
<td>Logical current drive - Default A:</td>
</tr>
<tr>
<td>PCurDrv</td>
<td>+2f7</td>
<td>1</td>
<td>B</td>
<td>Physical drive after assign mapping</td>
</tr>
<tr>
<td>Creating</td>
<td>+2f8</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>DelAll</td>
<td>+2f9</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FoundDel</td>
<td>+2fa</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Found_dev</td>
<td>+2fb</td>
<td>1</td>
<td>B</td>
<td>true =&gt; search found a device 3.10</td>
</tr>
<tr>
<td>fSplice</td>
<td>+2fc</td>
<td>1</td>
<td>B</td>
<td>true =&gt; do a splice in transpath 3.10</td>
</tr>
<tr>
<td>ClusFac</td>
<td>+2fd</td>
<td>1</td>
<td>B</td>
<td>sectors/cluster used in dir search</td>
</tr>
<tr>
<td>cMeta</td>
<td>+2fe</td>
<td>1</td>
<td>B</td>
<td>components found 3.10</td>
</tr>
<tr>
<td>PathNameType</td>
<td>+2ff</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>DevPt</td>
<td>+300</td>
<td>4</td>
<td>D</td>
<td>Address of device found by DevName <em>REDIR</em></td>
</tr>
<tr>
<td>DirSec</td>
<td>+304</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>DirStart</td>
<td>+308</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>NxtClusNum</td>
<td>+30a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>EntFree</td>
<td>+30c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>EntLast</td>
<td>+30e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>LastEnt</td>
<td>+310</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ProcFlag</td>
<td>+312</td>
<td>2</td>
<td>W</td>
<td>if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.</td>
</tr>
<tr>
<td>ptda_ForecedActions</td>
<td>+314</td>
<td>4</td>
<td>D</td>
<td>pending action bits</td>
</tr>
<tr>
<td>ptda_ulExitCode</td>
<td>+318</td>
<td>4</td>
<td>D</td>
<td>Exit code of last task</td>
</tr>
<tr>
<td>ptda_ulExitType</td>
<td>+31c</td>
<td>4</td>
<td>D</td>
<td>Type of exit</td>
</tr>
<tr>
<td>ptda_ulExitTID</td>
<td>+320</td>
<td>4</td>
<td>D</td>
<td>Exit Thread ID (32-bit exceptions)</td>
</tr>
<tr>
<td>ThisCDS</td>
<td>+324</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ptda_pCDS</td>
<td>+328</td>
<td>2</td>
<td>W</td>
<td>SS relative pointer to a curdir struct</td>
</tr>
<tr>
<td>CDSSize</td>
<td>+32a</td>
<td>2</td>
<td>W</td>
<td>Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg</td>
</tr>
<tr>
<td>Sattrib</td>
<td>+32c</td>
<td>2</td>
<td>W</td>
<td>Storage for search attrs <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>sPCB</td>
<td>+32e</td>
<td>2</td>
<td>W</td>
<td>Selector of Profile Control Block</td>
</tr>
<tr>
<td>ptda_pPCB</td>
<td>+330</td>
<td>4</td>
<td>D</td>
<td>Pointer to Profile Control Block</td>
</tr>
<tr>
<td>JFN_Max</td>
<td>+334</td>
<td>2</td>
<td>W</td>
<td>highest JFN used so far</td>
</tr>
<tr>
<td>NextSrchH</td>
<td>+336</td>
<td>2</td>
<td>W</td>
<td>Next value to use for search handle First value used will be 2.</td>
</tr>
</tbody>
</table>
SrchRmp +338 4 D Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First +33c 2 W
FNotifyLocal_Count +33e 2 W
Sig_ignf +340 2 W bit vector of ignored signals
Sig_hndf +342 2 W bit vector of handled signals
Sig_errf +344 2 W bit vector of error generating signals
Sig_attempted +346 2 W bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg +348 10 W byte vector of signal arguments
Sig_termtid +358 2 W 'Terminator' TID for APTERM.
HoldSigCnt +35a 2 W DOSHOLD SIGNAL counter
SigFocusCnt +35c 2 W PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table +35e 28 W default handle table
JFN_Flags +386 14 B default JFN flags table
ptda_rasflag +39a 2 W RAS trace indicator
SysSemPTDATbl +39c 100 S
SavedHardErr +49c 4 D
ptda_ptdasem +4a0 C S PTDA semaphore that is, inter-thread
ptda_DLMem +4ac C S b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt +4b8 6 W current IDT limit/base
Csid +4be 2 W Command Subtree ID
Behav_bit +4c0 2 W program behavior bits
MSW +4c2 2 W CPU matching status word
ptda_rsrlclist +4c4 4 D far pointer to local resource list
ptda_pldrdldHead +4c8 4 D loader demand load data list
pPrSemTbl +4cc 4 D (void * => PSEM) pointer to private semaphore table
ulPrTblSize +4d0 4 D size of pPrSemTbl in dwords
ulPrTotUsed +4d4 4 D number of entries in pPrSemTbl
ulPrNextFree +4d8 4 D next free slot in pPrSemTbl
hksPrTbl +4dc 4 D kernel semaphore handle for private semaphore table
pShSemBmp +4e0 4 D pointer to private bitmap for the shared semaphore table
ulShBmpSize +4e4 4 D size of pShSemBmp in bits
hksShBmp +4e8 4 D kernel semaphore handle for private semaphore table
ulMtxOwned +4ec 4 D number of mutex owned by this process in the two sem tables
ShareRetriesLeft +4f0 2 W number of share/lock viol retries
RetryCount +4f2 2 W num of share/lock retries to do
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RetryLoop</td>
<td>+4f4</td>
<td>2</td>
<td>W</td>
<td>num of share/lock retry delay loops ceb 75871</td>
</tr>
<tr>
<td>ptda_pSrchBuf</td>
<td>+4f6</td>
<td>2</td>
<td>W</td>
<td>internal search buffer</td>
</tr>
<tr>
<td>ptda_pad1</td>
<td>+4f8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_pOpenBuf</td>
<td>+4fa</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_TLMA</td>
<td>+4fc</td>
<td>4</td>
<td>D</td>
<td>in use flag and dword copy count</td>
</tr>
<tr>
<td>ptda_TLMABM</td>
<td>+500</td>
<td>4</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>ptda_TLMASizeMap</td>
<td>+504</td>
<td>20</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>Cons_Loc</td>
<td>+524</td>
<td>A</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SysCallSfcn</td>
<td>+52e</td>
<td>1</td>
<td>B</td>
<td>Value of AL on system entry</td>
</tr>
<tr>
<td>SysCall</td>
<td>+52f</td>
<td>1</td>
<td>B</td>
<td>Last system call processed</td>
</tr>
<tr>
<td>KBD_Mode</td>
<td>+530</td>
<td>1</td>
<td>B</td>
<td>Keyboard input mode</td>
</tr>
<tr>
<td>ptda_NewFiles</td>
<td>+531</td>
<td>1</td>
<td>B</td>
<td>If bit one is set, process supports // 54400 new files (long names)</td>
</tr>
<tr>
<td>AutoFail</td>
<td>+532</td>
<td>1</td>
<td>B</td>
<td>Non-zero if I 24 FAILED magically</td>
</tr>
<tr>
<td>ptda_direntry</td>
<td>+533</td>
<td>20</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>CP_Flags</td>
<td>+553</td>
<td>1</td>
<td>B</td>
<td>Default is no codepage in system.</td>
</tr>
<tr>
<td>Sig_vec</td>
<td>+554</td>
<td>20</td>
<td>D</td>
<td>signal handlers</td>
</tr>
<tr>
<td>Exc_vec</td>
<td>+574</td>
<td>1C</td>
<td>D</td>
<td>OSOLETE exception vectors</td>
</tr>
<tr>
<td>ptda_timerhead</td>
<td>+590</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Attrib</td>
<td>+594</td>
<td>2</td>
<td>W</td>
<td>storage for file attributes <em>REDIR</em></td>
</tr>
<tr>
<td>ExtFCB</td>
<td>+596</td>
<td>1</td>
<td>B</td>
<td>Extended FCB</td>
</tr>
<tr>
<td>ptda_extsig</td>
<td>+597</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>ptda_lanman_sec</td>
<td>+598</td>
<td>4</td>
<td>D</td>
<td>Used by LANMAN &amp; HPFS for security.</td>
</tr>
<tr>
<td>ptda_pad2</td>
<td>+59c</td>
<td>2</td>
<td>W</td>
<td>alignment</td>
</tr>
<tr>
<td>ptda_ppgdata</td>
<td>+59e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_child</td>
<td>+5a0</td>
<td>2</td>
<td>W</td>
<td>New child PTDA handle (Child being Exec'ed)</td>
</tr>
<tr>
<td>ptda_chiddallas</td>
<td>+5a2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_handle</td>
<td>+5a4</td>
<td>2</td>
<td>W</td>
<td>handle to this segment</td>
</tr>
<tr>
<td>ptda_module</td>
<td>+5a6</td>
<td>2</td>
<td>W</td>
<td>program module handle for process</td>
</tr>
<tr>
<td>ptda_ldthandle</td>
<td>+5a8</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_ldtpgmap</td>
<td>+5aa</td>
<td>2</td>
<td>W</td>
<td>Bitmap of valid LDT pages</td>
</tr>
<tr>
<td>ptda_ldtaddr</td>
<td>+5ac</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>CP_CaseMapTbl</td>
<td>+5b0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>codepage_tag</td>
<td>+5b4</td>
<td>2</td>
<td>W</td>
<td>the current code page</td>
</tr>
<tr>
<td>JFN_Length</td>
<td>+5b6</td>
<td>2</td>
<td>W</td>
<td>Size of JFN table in bytes</td>
</tr>
<tr>
<td>JFN_pTable</td>
<td>+5b8</td>
<td>4</td>
<td>D</td>
<td>PM pointer to JFN table</td>
</tr>
<tr>
<td>JFN_Flg_Ptr</td>
<td>+5bc</td>
<td>4</td>
<td>D</td>
<td>pointer to JFN flags</td>
</tr>
<tr>
<td>Joins</td>
<td>+5c0</td>
<td>1</td>
<td>B</td>
<td>number of joins</td>
</tr>
<tr>
<td>ExtErr_Locus</td>
<td>+5c1</td>
<td>1</td>
<td>B</td>
<td>Extended Error Locus <em>REDIR</em> 3.10</td>
</tr>
</tbody>
</table>
### Per-Task Data Area for OS/2 Warp V3.0 RETAIL kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDAParent</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Parent PTDA</td>
</tr>
<tr>
<td>pPTDASelf</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>This PTDA</td>
</tr>
<tr>
<td>pPTDAFirstChild</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Head of child chain PTDA</td>
</tr>
<tr>
<td>pPTDAExecChild</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>New Child PTDA (Child being exec'ed)</td>
</tr>
<tr>
<td>pPTDANextSibling</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Next sibling's PTDA</td>
</tr>
<tr>
<td>pPTDAPrevSibling</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Previous sibling's PTDA</td>
</tr>
<tr>
<td>ptda_pszproc</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to the EXE file this process is executing. Used by PerfView</td>
</tr>
<tr>
<td>ptda_pTCBHole</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>some TCB before first Tid 'hole'</td>
</tr>
<tr>
<td>ptda_pTCBHead</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Head of list of active TCBs owned by this process</td>
</tr>
<tr>
<td>ptda_cTCB</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Number of TCBs in use</td>
</tr>
<tr>
<td>ptda_cltib</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td>Count of TIBs allocated</td>
</tr>
<tr>
<td>ptda_avatib</td>
<td>+28</td>
<td>10</td>
<td>D</td>
<td>Pointers to TIB arrays</td>
</tr>
<tr>
<td>ptda_pdcbb</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_flDbg</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_sh</td>
<td>+40</td>
<td>40</td>
<td>S</td>
<td>Private arena header</td>
</tr>
<tr>
<td>ptda_pgdata</td>
<td>+a8</td>
<td>26</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_environ</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td>handle to process's envt seg</td>
</tr>
<tr>
<td>ptda_pBeginLIBPATH</td>
<td>+a8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
ptda_pEndLIBPATH +ac 4 D D75220- support dynamic libpath
ptda_pgpcc +b0 1E0 S
ptda_pPVDPrc +290 4 D
ptda_pSGSLlist +294 4 D
ptda_pexllist +298 4 D Flat pointer to exit list data
ptda_cdllterm +29c 4 D
WFP_Start +2a0 2 W TASKAREA offset for working string *REDIR*
Ren_WFP +2a2 2 W WFB pointer for rename destination *REDIR*
WFP_Path_End +2a4 2 W End of Path component of string.
Curr.Dir_End +2a6 2 W
CDS_Handle +2a8 34 W *REDIR*
OEMPtr +2dc 2 W
LIS_Fgnd +2de 1 B
FgndOnly +2df 1 B foreground only flag
ptda_pTCBCritSec +2e0 4 D TCB that did enter CritSec
ptda_pTCPriQCritSec +2e4 4 D TCBs awaiting CritSec wakeup
ptda_cCritSec +2e8 2 W Critical Section Count
CurrentPDB +2ea 2 W Currently active PDB (V86 segment)
DTAddr +2ec 4 D User's I/O transfer address *REDIR*
seltss +2f0 2 W
VolID +2f2 1 B !0 if vol ID found in dir search
NoSetDir +2f3 1 B If TRUE, do not set directory
SpaceFlag +2f4 1 B Embedded spaces allowed in FCB
VerFlg +2f5 1 B Initialize with verify off
LCurDrv +2f6 1 B Logical current drive - Default A:
PCurDrv +2f7 1 B physical drive after assign mapping
Creating +2f8 1 B
DelAll +2f9 1 B
FoundDel +2fa 1 B
Found_dev +2fb 1 B true => search found a device 3.10
fSplice +2fc 1 B true => do a splice in transpath 3.10
ClusFac +2fd 1 B sectors/cluster used in dir search
cMeta +2fe 1 B components found 3.10
PathNameType +2ff 1 B
DevPt +300 4 D Address of device found by DevName *REDIR*
DirSec +304 4 D
DirStart +308 2 W
NxtClusNum +30a 2 W
<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntFree</td>
<td>+30c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>EntLast</td>
<td>+30e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>LastEnt</td>
<td>+310</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ProcFlag</td>
<td>+312</td>
<td>2</td>
<td>W</td>
<td>if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.</td>
</tr>
<tr>
<td>ptda_ForcedActions</td>
<td>+314</td>
<td>4</td>
<td>D</td>
<td>pending action bits</td>
</tr>
<tr>
<td>ptda_ulExitCode</td>
<td>+318</td>
<td>4</td>
<td>D</td>
<td>Exit code of last task</td>
</tr>
<tr>
<td>ptda_ulExitType</td>
<td>+31c</td>
<td>4</td>
<td>D</td>
<td>Type of exit</td>
</tr>
<tr>
<td>ptda_ulExitTID</td>
<td>+320</td>
<td>4</td>
<td>D</td>
<td>Exit Thread ID (32-bit exceptions)</td>
</tr>
<tr>
<td>ThisCDS</td>
<td>+324</td>
<td>4</td>
<td>D</td>
<td>Address of current CDS <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ptda_pCDS</td>
<td>+328</td>
<td>2</td>
<td>W</td>
<td>SS relative pointer to a curdir struct</td>
</tr>
<tr>
<td>CDSsize</td>
<td>+32a</td>
<td>2</td>
<td>W</td>
<td>Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg</td>
</tr>
<tr>
<td>Sattrib</td>
<td>+32c</td>
<td>2</td>
<td>W</td>
<td>Storage for search attrs <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>sPCB</td>
<td>+32e</td>
<td>2</td>
<td>W</td>
<td>Selector of Profile Control Block</td>
</tr>
<tr>
<td>ptda_pPCB</td>
<td>+330</td>
<td>4</td>
<td>D</td>
<td>Pointer to Profile Control Block</td>
</tr>
<tr>
<td>JFN_Max</td>
<td>+334</td>
<td>2</td>
<td>W</td>
<td>highest JFN used so far</td>
</tr>
<tr>
<td>NextSrchH</td>
<td>+336</td>
<td>2</td>
<td>W</td>
<td>Next value to use for search handle First value used will be 2.</td>
</tr>
<tr>
<td>SrchRmp</td>
<td>+338</td>
<td>4</td>
<td>D</td>
<td>Handle &amp; Selector for RMP segment we keep search handles in.</td>
</tr>
<tr>
<td>FNotifyLocal_First</td>
<td>+33c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>FNotifyLocal_Count</td>
<td>+33e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>Sig_ignf</td>
<td>+340</td>
<td>2</td>
<td>W</td>
<td>bit vector of ignored signals</td>
</tr>
<tr>
<td>Sig_hndf</td>
<td>+342</td>
<td>2</td>
<td>W</td>
<td>bit vector of handled signals</td>
</tr>
<tr>
<td>Sig_errf</td>
<td>+344</td>
<td>2</td>
<td>W</td>
<td>bit vector of error generating signals</td>
</tr>
<tr>
<td>Sig_attempted</td>
<td>+346</td>
<td>2</td>
<td>W</td>
<td>bit vector of signals we've tried to handle with 32-bit exceptions</td>
</tr>
<tr>
<td>Sig_arg</td>
<td>+348</td>
<td>10</td>
<td>W</td>
<td>byte vector of signal arguments</td>
</tr>
<tr>
<td>Sig_terminid</td>
<td>+358</td>
<td>2</td>
<td>W</td>
<td>'Terminator' TID for ATERM.</td>
</tr>
<tr>
<td>HoldSigCnt</td>
<td>+35a</td>
<td>2</td>
<td>W</td>
<td>DOSHOLD SIGNAL counter</td>
</tr>
<tr>
<td>SigFocusCnt</td>
<td>+35c</td>
<td>2</td>
<td>W</td>
<td>PUBLIB DOS32SETSIGALEXCEPTIONFOCUS count</td>
</tr>
<tr>
<td>JFN_Table</td>
<td>+35e</td>
<td>28</td>
<td>W</td>
<td>default handle table</td>
</tr>
<tr>
<td>JFN_Flags</td>
<td>+386</td>
<td>14</td>
<td>B</td>
<td>default JFN flags table</td>
</tr>
<tr>
<td>ptda_rasflag</td>
<td>+39a</td>
<td>2</td>
<td>W</td>
<td>RAS trace indicator</td>
</tr>
<tr>
<td>SysSemPTDATbl</td>
<td>+39c</td>
<td>100</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SavedHardErr</td>
<td>+49c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_ptdasem</td>
<td>+4a0</td>
<td>8</td>
<td>S</td>
<td>PTDA semaphore that is, inter-thread</td>
</tr>
<tr>
<td>ptda_DLMssem</td>
<td>+4a8</td>
<td>8</td>
<td>S</td>
<td>b732954 Edd PTDA semaphore that is, inter-thread</td>
</tr>
<tr>
<td>ptda_lidt</td>
<td>+4b0</td>
<td>6</td>
<td>W</td>
<td>current IDT limit/base</td>
</tr>
<tr>
<td>Variable</td>
<td>Offset</td>
<td>Size</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CsId</td>
<td>+4b6</td>
<td>2</td>
<td>W</td>
<td>Command Subtree ID</td>
</tr>
<tr>
<td>Behav_bit</td>
<td>+4b8</td>
<td>2</td>
<td>W</td>
<td>program behavior bits</td>
</tr>
<tr>
<td>MSW</td>
<td>+4ba</td>
<td>2</td>
<td>W</td>
<td>CPU matching status word</td>
</tr>
<tr>
<td>ptda_rsrcclist</td>
<td>+4bc</td>
<td>4</td>
<td>D</td>
<td>far pointer to local resource list</td>
</tr>
<tr>
<td>ptda_plrdldHead</td>
<td>+4c0</td>
<td>4</td>
<td>D</td>
<td>loader demand load data list</td>
</tr>
<tr>
<td>pPrSemTbl</td>
<td>+4c4</td>
<td>4</td>
<td>D</td>
<td>(void * =&gt; PSEM) pointer to private semaphore table</td>
</tr>
<tr>
<td>ulPrTblSize</td>
<td>+4c8</td>
<td>4</td>
<td>D</td>
<td>size of pPrSemTbl in dwords</td>
</tr>
<tr>
<td>ulPrTotUsed</td>
<td>+4cc</td>
<td>4</td>
<td>D</td>
<td>number of entries in pPrSemTbl</td>
</tr>
<tr>
<td>ulPrNextFree</td>
<td>+4d0</td>
<td>4</td>
<td>D</td>
<td>next free slot in pPrSemTbl</td>
</tr>
<tr>
<td>hksPrTbl</td>
<td>+4d4</td>
<td>4</td>
<td>D</td>
<td>kernel semaphore handle for private semaphore table</td>
</tr>
<tr>
<td>pShSemBmp</td>
<td>+4d8</td>
<td>4</td>
<td>D</td>
<td>pointer to private bitmap for the shared semaphore table</td>
</tr>
<tr>
<td>ulShBmpSize</td>
<td>+4dc</td>
<td>4</td>
<td>D</td>
<td>size of pShSemBmp in bits</td>
</tr>
<tr>
<td>hksShBmp</td>
<td>+4e0</td>
<td>4</td>
<td>D</td>
<td>kernel semaphore handle for private semaphore table</td>
</tr>
<tr>
<td>ulMtxOwned</td>
<td>+4e4</td>
<td>4</td>
<td>D</td>
<td>number of mutex owned by this process in the two sem tables</td>
</tr>
<tr>
<td>ShareRetriesLeft</td>
<td>+4e8</td>
<td>2</td>
<td>W</td>
<td>number of share/lock viol retries</td>
</tr>
<tr>
<td>RetryCount</td>
<td>+4ea</td>
<td>2</td>
<td>W</td>
<td>num of share/lock retries to do</td>
</tr>
<tr>
<td>RetryLoop</td>
<td>+4ec</td>
<td>2</td>
<td>W</td>
<td>num of share/lock retry delay loops ceb 75871</td>
</tr>
<tr>
<td>ptda_pSrchBuf</td>
<td>+4ee</td>
<td>2</td>
<td>W</td>
<td>internal search buffer</td>
</tr>
<tr>
<td>ptda_pad1</td>
<td>+4f0</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_pOpenBuf</td>
<td>+4f2</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_TLMA</td>
<td>+4f4</td>
<td>4</td>
<td>D</td>
<td>in use flag and dword copy count</td>
</tr>
<tr>
<td>ptda_TLMABM</td>
<td>+4f8</td>
<td>4</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>ptda_TLMASizeMap</td>
<td>+4fc</td>
<td>20</td>
<td>B</td>
<td>thread local memory</td>
</tr>
<tr>
<td>Cons_Loc</td>
<td>+51c</td>
<td>A</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SysCallSfcn</td>
<td>+526</td>
<td>1</td>
<td>B</td>
<td>Value of AL on system entry</td>
</tr>
<tr>
<td>SysCall</td>
<td>+527</td>
<td>1</td>
<td>B</td>
<td>Last system call processed</td>
</tr>
<tr>
<td>KBD_Mode</td>
<td>+528</td>
<td>1</td>
<td>B</td>
<td>Keyboard input mode</td>
</tr>
<tr>
<td>ptda_NewFiles</td>
<td>+529</td>
<td>1</td>
<td>B</td>
<td>If bit one is set, process supports 54400 new files (long names)</td>
</tr>
<tr>
<td>AutoFail</td>
<td>+52a</td>
<td>1</td>
<td>B</td>
<td>Non-zero if I 24 FAILED magically</td>
</tr>
<tr>
<td>ptda_direntry</td>
<td>+52b</td>
<td>20</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>CP_Flgs</td>
<td>+54b</td>
<td>1</td>
<td>B</td>
<td>Default is no codepage in system.</td>
</tr>
<tr>
<td>Sig_vec</td>
<td>+54c</td>
<td>20</td>
<td>D</td>
<td>signal handlers</td>
</tr>
<tr>
<td>Exc_vec</td>
<td>+56c</td>
<td>1C</td>
<td>D</td>
<td>OSELETE exception vectors</td>
</tr>
<tr>
<td>ptda_timerhead</td>
<td>+588</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Attrib</td>
<td>+58c</td>
<td>2</td>
<td>W</td>
<td>storage for file attributes <em>REDIR</em></td>
</tr>
<tr>
<td>ExtFCB</td>
<td>+58e</td>
<td>1</td>
<td>B</td>
<td>Extended FCB</td>
</tr>
</tbody>
</table>
Per-Task Data Area for OS/2 V2.11 ALLSTRICT kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset Length Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDAParent</td>
<td>+0 4 D</td>
<td>Parent PTDA</td>
</tr>
</tbody>
</table>

ptda_extsig           +58f  1  B     
ptda_lanman_sec        +590  4  D     Used by LANMAN & HPFS for security.
ptda_pad2              +594  2  W     alignment
ptda_ppgdata           +596  2  W
ptda_child             +598  2  W     New child PTDA handle (Child being Exec'ed)
ptda_childalias        +59a  2  W
ptda_handle            +59c  2  W     handle to this segment
ptda_module            +59e  2  W     program module handle for process
ptda_ldthandle         +5a0  2  W
ptda_ldtpgmap          +5a2  2  W     Bitmap of valid LDT pages
ptda_ldtaddr           +5a4  4  D
CP_CaseMapTbl          +5a8  4  D
codepage_tag           +5ac  2  W     the current code page
JFN_Length             +5ae  2  W     Size of JFN table in bytes
JFN_pTable             +5b0  4  D     PM pointer to JFN table
JFN_Flg_Ptr            +5b4  4  D     pointer to JFN flags
Joins                  +5b8  1  B     number of joins
ExtErr_Locus           +5b9  1  B     Extended Error Locus *REDIR* 3.10
ExtErr                 +5ba  2  W     Extended Error code *REDIR* 3.10
ExtErr_Action          +5bc  1  B     Extended Error Action *REDIR* 3.10
ExtErr_Class           +5bd  1  B     Extended Error Class *REDIR* 3.10
ptda_infoseg           +5be  24  S
ptda_pad3              +5e2  2  W     alignment
CurrTCB                +5e4  2  W     pointer to current TCB
CurrTSD                +5e6  2  W     pointer to current TSD
ThisPTDA               +5e8  2  W     Selector for this ptda
ptda_NPX_em_cs         +5ea  2  W     b726833 NPX emulator CS b726833
ptda_NPX_em_eip        +5ec  4  D     b726833 NPX emulator EIP b726833
ptda_pad4              +5f0  2  W     alignment b726833
ptda_signature         +5f2  2  B     must contain "TD"
<table>
<thead>
<tr>
<th>Variable</th>
<th>Offset</th>
<th>Size</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pPTDASelf</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>This PTDA</td>
</tr>
<tr>
<td>pPTDAFirstChild</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Head of child chain PTDA</td>
</tr>
<tr>
<td>pPTDAExecChild</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>New Child PTDA (Child being exec'ed)</td>
</tr>
<tr>
<td>pPTDNSibling</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Next sibling's PTDA</td>
</tr>
<tr>
<td>pPTDAPrevSibling</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Previous sibling's PTDA</td>
</tr>
<tr>
<td>ptda_pszproc</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to the EXE file this process is executing. Used by PerfView</td>
</tr>
<tr>
<td>ptda_pTCBHole</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>some TCB before first Tid 'hole'</td>
</tr>
<tr>
<td>ptda_pTCBHead</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Head of list of active TCBs owned by this process</td>
</tr>
<tr>
<td>ptda_cTCB</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td>Number of TCBs in use</td>
</tr>
<tr>
<td>ptda_ctib</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td>Count of TIBs allocated</td>
</tr>
<tr>
<td>ptda_avatib</td>
<td>+28</td>
<td>10</td>
<td>D</td>
<td>Pointers to TIB arrays</td>
</tr>
<tr>
<td>ptda_pdcb</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_flDbg</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_ah</td>
<td>+40</td>
<td>40</td>
<td>S</td>
<td>Private arena header</td>
</tr>
<tr>
<td>ptda_pgdata</td>
<td>+80</td>
<td>26</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_environ</td>
<td>+a6</td>
<td>2</td>
<td>W</td>
<td>handle to process's envt seg</td>
</tr>
<tr>
<td>ptda_pppc</td>
<td>+a8</td>
<td>400</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pmemstatcur</td>
<td>+4a8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_memstat</td>
<td>+4ac</td>
<td>3C</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pPVDIBPrC</td>
<td>+4e8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pSGSLList</td>
<td>+4ec</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ptda_pexllist</td>
<td>+f0</td>
<td>4</td>
<td>D</td>
<td>Flat pointer to exit list data</td>
</tr>
<tr>
<td>ptda_cdllterm</td>
<td>+f4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>WFP_Start</td>
<td>+f8</td>
<td>2</td>
<td>W</td>
<td>TASKAREA offset for working string <em>REDIR</em></td>
</tr>
<tr>
<td>Ren_WFP</td>
<td>+fa</td>
<td>2</td>
<td>W</td>
<td>WFB pointer for rename destination <em>REDIR</em></td>
</tr>
<tr>
<td>WFP_Path_End</td>
<td>+fc</td>
<td>2</td>
<td>W</td>
<td>End of Path component of string.</td>
</tr>
<tr>
<td>Curr_Dir_End</td>
<td>+fe</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>CDS_Handle</td>
<td>+500</td>
<td>34</td>
<td>W</td>
<td><em>REDIR</em></td>
</tr>
<tr>
<td>OEMPtr</td>
<td>+534</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>LIS_Fgnd</td>
<td>+536</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FgndOnly</td>
<td>+537</td>
<td>1</td>
<td>B</td>
<td>foreground only flag</td>
</tr>
<tr>
<td>ptda_pTCBCritSec</td>
<td>+538</td>
<td>4</td>
<td>D</td>
<td>TCB that did enter CritSec</td>
</tr>
<tr>
<td>ptda_pTCPriQCritSec</td>
<td>+53c</td>
<td>4</td>
<td>D</td>
<td>TCBs awaiting CritSec wakeup</td>
</tr>
<tr>
<td>ptda_cCritSec</td>
<td>+540</td>
<td>2</td>
<td>W</td>
<td>Critical Section Count</td>
</tr>
<tr>
<td>CurrentPDB</td>
<td>+542</td>
<td>2</td>
<td>W</td>
<td>Currently active PDB (V86 segment)</td>
</tr>
<tr>
<td>DTAddr</td>
<td>+544</td>
<td>4</td>
<td>D</td>
<td>User's I/O transfer address <em>REDIR</em></td>
</tr>
<tr>
<td>seltss</td>
<td>+548</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
VolID                +54a   1      B     10 if vol ID found in dir search
NoSetDir             +54b   1      B     If TRUE, do not set directory
SpaceFlag            +54c   1      B     Embedded spaces allowed in FCB
VerFlg               +54d   1      B     Initialize with verify off
LCurDrv              +54e   1      B     Logical current drive - Default A:
PCurDrv              +54f   1      B     physical drive after assign mapping
Creating             +550   1      B
DelAll               +551   1      B
FoundDel             +552   1      B
Found_dev            +553   1      B     true => search found a device 3.10
fSplice              +554   1      B     true => do a splice in transpath 3.10
ClusFac              +555   1      B     sectors/cluster used in dir search
cMeta                +556   1      B     components found 3.10
PathNameType         +557   1      B
DevPt                +558   4      D     Address of device found by DevName
*REDIR*
DirSec               +55c   4      D
DirStart             +560   2      W
NxtClusNum           +562   2      W
EntFree              +564   2      W
EntLast              +566   2      W
LastEnt              +568   2      W
ProcFlag             +56a   2      W     if == 1 then this is a special process
                                        (swapper or screen switch); NO removable
                                        media buffer will be allocated to this
                                        process.
ptda_ForcedActions   +56c   4      D     pending action bits
ptda_ulExitCode      +570   4      D     Exit code of last task
ptda_ulExitType      +574   4      D     Type of exit
ptda_ulExitTID       +578   4      D     Exit Thread ID (32-bit exceptions)
ThisCDS              +57c   4      D     Address of current CDS *REDIR* 3.10
ptda_pCDS            +580   2      W    SS relative pointer to a curdir struct
CDSSize              +582   2      W     Size of CDS pointed to by ThisCDS ONLY
                                        used for CDS entries in RMP seg
Sattrib              +584   2      W     Storage for search attrs *REDIR* 3.10
sPCB                 +586   2      W    Selector of Profile Control Block
ptda_pPCB            +588   4      D     Pointer to Profile Control Block
JFN_Max              +58c   2      W     highest JFN used so far
NextSrchH            +58e   2      W     Next value to use for search handle
                                        First value used will be 2.
Srchrmp               +590   4      D     Handle & Selector for RMP segment we
                                        keep search handles in.
FNotifyLocal_First   +594   2      W
FNotifyLocal_Count +596 2 W
Sig_ignf +598 2 W bit vector of ignored signals
Sig_hndf +59a 2 W bit vector of handled signals
Sig_errf +59c 2 W bit vector of error generating signals
Sig_attempted +59e 2 W bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg +5a0 10 W byte vector of signal arguments
Sig_termtid +5b0 2 W 'Terminator' TID for APTERM.
HoldSigCnt +5b2 2 W DOSHoldsSignal counter
SigFocusCnt +5b4 2 W PUBLIB DOS32SERTYSISNALEXCEPTIONFOCUS count
JFN_Table +5b6 28 W default handle table
JFN_Flags +5de 14 B default JFN flags table
ptda_rasflag +5f2 2 W RAS trace indicator
SysSemPTDATbl +5f4 100 S
SavedHardErr +6f4 4 D
ptda_ptdasem +6f8 C S PTDA semaphore that is, inter-thread
ptda_DLMsem +704 C S b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt +710 6 W current IDT limit/base
Csid +716 2 W Command Subtree ID
Behav_bit +718 2 W program behavior bits
MSW +71a 2 W CPU matching status word
ptda_rsrclist +71c 4 D far pointer to local resource list
ptda_pldrdldHead +720 4 D loader demand load data list
pPrSemTbl +724 4 D (void * => PSEM) pointer to private semaphore table
ulPrTblSize +728 4 D size of pPrSemTbl in dwords
ulPrTotUsed +72c 4 D number of entries in pPrSemTbl
ulPrNextFree +730 4 D next free slot in pPrSemTbl
hksPrTbl +734 4 D kernel semaphore handle for private semaphore table
pShSemBmp +738 4 D pointer to private bitmap for the shared semaphore table
ulShBmpSize +73c 4 D size of pShSemBmp in bits
hksShBmp +740 4 D kernel semaphore handle for private semaphore table
ulMtxOwned +744 4 D number of mutex owned by this process in the two sem tables
ShareRetriesLeft +748 2 W number of share/lock viol retries
RetryCount +74a 2 W num of share/lock retries to do
ptda_pSrchBuf +74e 2 W internal search buffer
ptda_LibiError +750 2 W reuse same field to hold library init
errors

ptda_pOpenBuf +752 2 W
Cons_Loc +754 A S
SysCallSfcn +75e 1 B Value of AL on system entry
SysCall +75f 1 B Last system call processed
KBD_Mode +760 1 B Keyboard input mode
ptda_NewFiles +761 1 B If bit one is set, process supports // 54400 new files (long names)
AutoFail +762 1 B Non-zero if I 24 FAILED magically
ptda_direntry +763 20 S
CP_Flags +783 1 B Default is no codepage in system.
Sig_vec +784 20 D signal handlers
Exc_vec +7a4 1C D OSOLETE exception vectors
ptda_timerhead +7c0 4 D
Attrib +7c4 2 W storage for file attributes *REDIR*
ExtFCB +7c6 1 B Extended FCB
ptda_extsig +7c7 1 B
ptda_lanman_sec +7c8 4 D Used by LANMAN & HPFS for security.
ptda_pad2 +7cc 2 W alignment
ptda_pppdata +7ce 2 W
ptda_child +7d0 2 W New child PTDA handle (Child being Exec'ed)
ptda_childalias +7d2 2 W
ptda_handle +7d4 2 W handle to this segment
ptda_module +7d6 2 W program module handle for process
ptda_ldthandle +7d8 2 W
ptda_ldtpgmap +7da 2 W Bitmap of valid LDT pages
ptda_ldtaddr +7dc 4 D
CP_CaseMapTbl +7e0 4 D
codepage_tag +7e4 2 W the current code page
JFN_Length +7e6 2 W Size of JFN table in bytes
JFN_pTable +7e8 4 D PM pointer to JFN table
JFN_Flag_Ptr +7ec 4 D pointer to JFN flags
Joins +7f0 1 B number of joins
ExtErr_Locus +7f1 1 B Extended Error Locus *REDIR* 3.10
ExtErr +7f2 2 W Extended Error code *REDIR* 3.10
ExtErr_Action +7f4 1 B Extended Error Action *REDIR* 3.10
ExtErr_Class +7f5 1 B Extended Error Class *REDIR* 3.10
ptda_infoseg +7f6 24 S
ptda_pad3 +81a 2 W alignment
CurrTCB +81c 2 W pointer to current TCB
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrTSD</td>
<td>+01e</td>
<td>2</td>
<td>W</td>
<td>pointer to current TSD</td>
</tr>
<tr>
<td>ThisPTDA</td>
<td>+820</td>
<td>2</td>
<td>W</td>
<td>Selector for this ptda</td>
</tr>
<tr>
<td>ptda_NPX_em_cs</td>
<td>+822</td>
<td>2</td>
<td>W</td>
<td>b726833 NPX emulator CS b726833</td>
</tr>
<tr>
<td>ptda_NPX_em_eip</td>
<td>+824</td>
<td>4</td>
<td>D</td>
<td>b726833 NPX emulator EIP b726833</td>
</tr>
<tr>
<td>ptda_pad4</td>
<td>+828</td>
<td>2</td>
<td>W</td>
<td>alignment b726833</td>
</tr>
<tr>
<td>ptda_signature</td>
<td>+82a</td>
<td>2</td>
<td>B</td>
<td>must contain &quot;TD&quot;</td>
</tr>
</tbody>
</table>

Per-Task Data Area for OS/2 V2.11 RETAIL kernel

Field Name                      Offset | Length | Type | Description                                                                 |
---------------------------------+--------|--------|------|-----------------------------------------------------------------------------|
| pPTDAParent                     | +0     | 4      | D    | Parent PTDA                                                                |
| pPTDASelf                       | +4     | 4      | D    | This PTDA                                                                  |
| pPTDAFirstChild                 | +8     | 4      | D    | Head of child chain PTDA                                                  |
| pPTDAExecChild                  | +c     | 4      | D    | New Child PTDA (Child being exec'ed)                                      |
| pPTDANextSibling                | +10    | 4      | D    | Next sibling's PTDA                                                       |
| pPTDAPrevSibling                | +14    | 4      | D    | Previous sibling's PTDA                                                   |
| ptda_pszproc                   | +18    | 4      | D    | Pointer to the EXE file this process is executing. Used by PerfView       |
| ptda_pTCBHole                  | +1c    | 4      | D    | some TCB before first Tid 'hole'                                          |
| ptda_pTCBHead                  | +20    | 4      | D    | Head of list of active TCBs owned by this process                        |
| ptda_cTCB                      | +24    | 2      | W    | Number of TCBs in use                                                     |
| ptda_ctib                      | +26    | 2      | W    | Count of TIBs allocated                                                   |
| ptda_avatib                    | +28    | 10     | D    | Pointers to TIB arrays                                                   |
| ptda_pdcb                      | +38    | 4      | D    |                                                                               |
| ptda_f1Dbg                     | +3c    | 4      | D    |                                                                               |
| ptda_sh                        | +40    | 40     | S    | Private arena header                                                      |
| ptda_pgdata                    | +80    | 26     | S    |                                                                               |
| ptda_environ                  | +a6    | 2      | W    | handle to process's envt seg                                              |
| ptda_ppgc                      | +a8    | 400    | S    |                                                                               |
| ptda_pmemstatcur              | +4a8   | 4      | D    |                                                                               |
| ptda_memstat                  | +4ac   | 3C     | S    |                                                                               |
| ptda_pFVDBPrc                 | +4e8   | 4      | D    |                                                                               |
| ptda_pSGSList                 | +4ec   | 4      | D    |                                                                               |
| ptda_pexllist                 | +4f0   | 4      | D    | Flat pointer to exit list data                                            |
| ptda_cdilterm                 | +4f4   | 4      | D    |                                                                               |
WFP_Start +4f8  2 W TASKAREA offset for working string
               *REDIR*
Ren_WFP          +4fa  2 W WFB pointer for rename destination
               *REDIR*
WFP_Path_End    +4fc  2 W End of Path component of string.
Curr_Dir_End     +4fe  2 W
CDS_Handle       +500  34 W *REDIR*
OEMPtr           +534  2 W
LIS_Fgnd         +536  1 B
FgndOnly         +537  1 B foreground only flag
ptda_pTCBCritSec +538  4 D TCB that did enter CritSec
ptda_pTCB PriQCritSec +53c  4 D TCBs awaiting CritSec wakeup
ptda_cCritSec    +540  2 W Critical Section Count
CurrentPDB       +542  2 W Currently active PDB (V86 segment)
DTAddr           +544  4 D User's I/O transfer address *REDIR*
seltss           +548  2 W
VolID            +54a  1 B 0 if vol ID found in dir search
NoSetDir         +54b  1 B If TRUE, do not set directory
SpaceFlag        +54c  1 B Embedded spaces allowed in FCB
VerFlg           +54d  1 B Initialize with verify off
LCurDrv          +54e  1 B Logical current drive – Default A:
PCurDrv          +54f  1 B physical drive after assign mapping
Creating         +550  1 B
DelAll           +551  1 B
FoundDel         +552  1 B
Found_dev        +553  1 B true => search found a device 3.10
fSplice          +554  1 B true => do a splice in transpath 3.10
ClusFac          +555  1 B sectors/cluster used in dir search
cMeta            +556  1 B components found 3.10
PathNameType     +557  1 B
DevPt            +558  4 D Address of device found by DevName
               *REDIR*
DirSec           +55c  4 D
DirStart         +560  2 W
NxtClusNum       +562  2 W
EntFree          +564  2 W
EntLast          +566  2 W
LastEnt          +568  2 W
ProcFlag         +56a  2 W if == 1 then this is a special process
               (swapper or screen switch); NO removable
               media buffer will be allocated to this process.
ptda_ForcedActions   +56c   4      D     pending action bits
ptda_ulExitCode      +570   4      D     Exit code of last task
ptda_ulExitType      +574   4      D     Type of exit
ptda_ulExitTID       +578   4      D     Exit Thread ID (32-bit exceptions)
ThisCDS              +57c   4      D     Address of current CDS *REDIR* 3.10
ptda_pCDS            +580   2      W    SS relative pointer to a curdir struct
CDSsize              +582   2      W     Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib              +584   2      W     Storage for search attrs *REDIR* 3.10
sPCB                 +586   2      W    Selector of Profile Control Block
ptda_pPCB            +588   4      D    Pointer to Profile Control Block
JFN_Mat              +58c   2      W     highest JFN used so far
NextSrchH            +58e   2      W    Next value to use for search handle First value used will be 2.
SrchRmp              +590   4      D     Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First   +594   2      W
FNotifyLocal_Count   +596   2      W
Sig_ignf             +598   2      W     bit vector of ignored signals
Sig_hndf             +59a   2      W     bit vector of handled signals
Sig_errf             +59c   2      W     bit vector of error generating signals
Sig_attempted        +59e   2      W     bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg              +5a0   10     W    byte vector of signal arguments
Sig_termid           +5b0   2      W     'Terminator' TID for ATERM.
HoldSigCnt           +5b2   2      W     DOSHOLDSIGNAL counter
SigFocusCnt          +5b4   2      W    PUBLIB DOS32SETSIGALEXCEPTIONFOCUS count
JFN_Table            +5b6   28     W     default handle table
JFN_Flags            +5de  14     B     default JFN flags table
ptda_rasflag         +5f2   2      W     RAS trace indicator
SysSemPTDATbl         +5f4  100     S
SavedHardErr         +6f4   4      D
ptda_ptdasem         +6f8   8      S    PTDA semaphore that is, inter-thread
ptda_DLMsem          +700   8      S    b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt            +708   6      W     current IDT limit/base
Csid                 +70e   2      W     Command Subtree ID
Behav_bit            +710   2      W    program behavior bits
MSW                  +712   2      W    CPU matching status word
ptda_rsrclist        +714   4      D     far pointer to local resource list
ptda_pldrdldHead     +718   4      D    loader demand load data list
pPrSemTbl            +71c   4      D     (void * => PSEM) pointer to private
semaphore table

ulPrTblSize          +720   4      D     size of pPrSemTbl in dwords
ulPrTotUsed          +724   4      D     number of entries in pPrSemTbl
ulPrNextFree         +728   4      D     next free slot in pPrSemTbl
hksPrTbl             +72c   4      D     kernel semaphore handle for private semaphore table
pShSemBmp            +730   4      D     pointer to private bitmap for the shared semaphore table
ulShBmpSize          +734   4      D     size of pShSemBmp in bits
hksShBmp             +738   4      D     kernel semaphore handle for private semaphore table
ulMtxOwned           +73c   4      D     number of mutex owned by this process in the two sem tables
ShareRetriesLeft     +740   2      W    number of share/lock viol retries
RetryCount           +742   2      W    num of share/lock retries to do
RetryLoop            +744   2      W
ptda_pSrchBuf        +746   2      W    internal search buffer
ptda_LibiError       +748   2      W    reuse same field to hold library init errors
ptda_pOpenBuf        +74a   2      W
Cons_Loc             +74c   1      S
SysCallSfcn          +756   1      B     Value of AL on system entry
SysCall              +757   1      B     Last system call processed
KBD_Mode             +758   1      B     Keyboard input mode
ptda_NewFiles        +759   1      B    If bit one is set, process supports // 54400 new files (long names)
AutoFail             +75a   1      B    Non-zero if I 24 FAILED magically
ptda_direntry        +75b   20     S
CP_Flags             +77b   1      B    Default is no codepage in system.
Sig_vec              +77c   20     D     signal handlers
Exc_vec              +79c   1C     D     OSOLETE exception vectors
ptda_timerhead       +7b8   4      D
Attrib               +7bc   2      W    storage for file attributes *REDIR*
ExtFCB               +7be   1      B     Extended FCB
ptda_extsig          +7bf   1      B
ptda_lanman_sec      +7c0   4      D     Used by LANMAN & HPFS for security.
ptda_pad2            +7c4   2      W    alignment
ptda_ppgdata         +7c6   2      W
ptda_child           +7c8   2      W    New child PTDA handle (Child being Exec'ed)
ptda_childalias      +7ca   2      W
ptda_handle          +7cc   2      W    handle to this segment
ptda_module          +7ce   2      W    program module handle for process
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptda_ldthandle</td>
<td>+7d0</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ptda_ldtpgmap</td>
<td>+7d2</td>
<td>2</td>
<td>W</td>
<td>Bitmap of valid LDT pages</td>
</tr>
<tr>
<td>ptda_ltdaddr</td>
<td>+7d4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>CP_CaseMapTbl</td>
<td>+7d8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>codepage_tag</td>
<td>+7dc</td>
<td>2</td>
<td>W</td>
<td>the current code page</td>
</tr>
<tr>
<td>JFN_Length</td>
<td>+7de</td>
<td>2</td>
<td>W</td>
<td>Size of JFN table in bytes</td>
</tr>
<tr>
<td>JFN_pTable</td>
<td>+7e0</td>
<td>4</td>
<td>D</td>
<td>PM pointer to JFN table</td>
</tr>
<tr>
<td>JFN_Flg_Ptr</td>
<td>+7e4</td>
<td>4</td>
<td>D</td>
<td>pointer to JFN flags</td>
</tr>
<tr>
<td>Joins</td>
<td>+7e8</td>
<td>1</td>
<td>B</td>
<td>number of joins</td>
</tr>
<tr>
<td>ExtErr_Locus</td>
<td>+7e9</td>
<td>1</td>
<td>B</td>
<td>Extended Error Locus <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr</td>
<td>+7ea</td>
<td>2</td>
<td>W</td>
<td>Extended Error code <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr_Action</td>
<td>+7ec</td>
<td>1</td>
<td>B</td>
<td>Extended Error Action <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ExtErr_Class</td>
<td>+7ed</td>
<td>1</td>
<td>B</td>
<td>Extended Error Class <em>REDIR</em> 3.10</td>
</tr>
<tr>
<td>ptda_infoseg</td>
<td>+7ee</td>
<td>24</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>ptda_pad3</td>
<td>+812</td>
<td>2</td>
<td>W</td>
<td>alignment</td>
</tr>
<tr>
<td>CurrTCB</td>
<td>+814</td>
<td>2</td>
<td>W</td>
<td>pointer to current TCB</td>
</tr>
<tr>
<td>CurrTSD</td>
<td>+816</td>
<td>2</td>
<td>W</td>
<td>pointer to current TSD</td>
</tr>
<tr>
<td>ThisPTDA</td>
<td>+818</td>
<td>2</td>
<td>W</td>
<td>Selector for this ptda</td>
</tr>
<tr>
<td>ptda_NPX_em_cs</td>
<td>+81a</td>
<td>2</td>
<td>W</td>
<td>b726833 NPX emulator CS b726833</td>
</tr>
<tr>
<td>ptda_NPX_em_eip</td>
<td>+81c</td>
<td>4</td>
<td>D</td>
<td>b726833 NPX emulator EIP b726833</td>
</tr>
<tr>
<td>ptda_pad4</td>
<td>+820</td>
<td>2</td>
<td>W</td>
<td>alignment b726833</td>
</tr>
<tr>
<td>ptda_signature</td>
<td>+822</td>
<td>2</td>
<td>B</td>
<td>must contain &quot;TD&quot;</td>
</tr>
</tbody>
</table>

---------------------------------------------

Local Information Segment

Pointers
SAS field SAS_info_local points to the current LISEG.

Locations
dff:0 is the address of the copy of the LISEG for the current thread & process.
The LISEG for each process is imbedded in the PTDA at ptda_infoseg.

VM Owner
infoseg (0xff75)

Format
InfoSegLDT

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIS_CurProcID</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Current process ID</td>
</tr>
</tbody>
</table>
LIS_ParProcID +2 2 W Process ID of parent
LIS_CurThrdPri +4 2 W Current thread priority
LIS_CurThrdID +6 2 W Current thread ID
LIS_CurScrngrp +8 2 W Screengroup
LIS_ProcStatus +a 1 B Process status bits
LIS_fillbyte1 +b 1 B filler byte
LIS_Fgnd +c 2 W Current process is in foreground
LIS_ProcType +e 1 B Current process type
LIS_fillbyte2 +f 1 B filler byte
LIS_AX +10 2 W @@V1 Environment selector
LIS_BX +12 2 W @@V1 Offset of command line start
LIS_CX +14 2 W @@V1 Length of Data Segment
LIS_DX +16 2 W @@V1 STACKSIZE from the .EXE file
LIS_SI +18 2 W @@V1 HEAPSIZE from the .EXE file
LIS_DI +1a 2 W @@V1 Module handle of the application
LIS_DS +1c 2 W @@V1 Data Segment Handle of application
LIS_PackSel +1e 2 W First tiled selector in this EXE
LIS_PackShrSel +20 2 W First selector above shared arena
LIS_PackPckSel +22 2 W First selector above packed arena

**LIS_ProcStatus** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS_XITLST</td>
<td>0x01</td>
<td>Doing ExitList Processing</td>
</tr>
<tr>
<td>PS_XITH1</td>
<td>0x02</td>
<td>Exiting thread 1</td>
</tr>
<tr>
<td>PS_XITALL</td>
<td>0x04</td>
<td>The whole process is exiting</td>
</tr>
<tr>
<td>PS_SYNCPARENT</td>
<td>0x10</td>
<td>Parent cares about termination</td>
</tr>
<tr>
<td>PS_WAITPARENT</td>
<td>0x20</td>
<td>Parent did an exec-and-wait</td>
</tr>
<tr>
<td>PS_DYING</td>
<td>0x40</td>
<td>Process is dying</td>
</tr>
<tr>
<td>PS_EMBRYO</td>
<td>0x80</td>
<td>Process in embryonic state</td>
</tr>
</tbody>
</table>

**LIS_ProcType** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIS_PT_FULLSCRN</td>
<td>0</td>
<td>Full screen app.</td>
</tr>
<tr>
<td>LIS_PT_REALMODE</td>
<td>1</td>
<td>Real mode process</td>
</tr>
<tr>
<td>PT_VDM</td>
<td>1</td>
<td>VDM</td>
</tr>
<tr>
<td>LIS_PT_VIOWIN</td>
<td>2</td>
<td>VIO windowable app.</td>
</tr>
<tr>
<td>LIS_PT_PRESMGR</td>
<td>3</td>
<td>Presentation Manager app.</td>
</tr>
<tr>
<td>LIS_PT_DETACHED</td>
<td>4</td>
<td>Detached app.</td>
</tr>
</tbody>
</table>
Global Information Segment

Pointers
SAS field SAS_info_global points to the current GISEG.

Locations
df4:0 is the address of the copy of the GISEG for the current thread & process.

VM Owner
infoseg (0xff75) - shared arena copy
os2krnl (0xffaa) - system arena copy

Format
InfoSegGDT

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIS_BigTime</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Time from 1-1-1970 in seconds</td>
</tr>
<tr>
<td>SIS_MsCount</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Freerunning milliseconds counter</td>
</tr>
<tr>
<td>SIS_HrsTime</td>
<td>+8</td>
<td>1</td>
<td>B</td>
<td>Hours</td>
</tr>
<tr>
<td>SIS_MinTime</td>
<td>+9</td>
<td>1</td>
<td>B</td>
<td>Minutes</td>
</tr>
<tr>
<td>SIS_SecTime</td>
<td>+a</td>
<td>1</td>
<td>B</td>
<td>Seconds</td>
</tr>
<tr>
<td>SIS_HunTime</td>
<td>+b</td>
<td>1</td>
<td>B</td>
<td>Hundredths of seconds</td>
</tr>
<tr>
<td>SIS_TimeZone</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Timezone in min from GMT (Set to EST)</td>
</tr>
<tr>
<td>SIS_CLKInterval</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>Timer interval (units=0.0001 secs)</td>
</tr>
<tr>
<td>SIS_DayDate</td>
<td>+10</td>
<td>1</td>
<td>B</td>
<td>Day-of-month (1-31)</td>
</tr>
<tr>
<td>SIS_MonDate</td>
<td>+11</td>
<td>1</td>
<td>B</td>
<td>Month (1-12)</td>
</tr>
<tr>
<td>SIS_YrsDate</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>Year (&gt;= 1980)</td>
</tr>
<tr>
<td>SIS_DOWDate</td>
<td>+14</td>
<td>1</td>
<td>B</td>
<td>Day-of-week (1-1-80 = Tues = 3)</td>
</tr>
<tr>
<td>SIS_VerMajor</td>
<td>+15</td>
<td>1</td>
<td>B</td>
<td>Major version number</td>
</tr>
<tr>
<td>SIS_VerMinor</td>
<td>+16</td>
<td>1</td>
<td>B</td>
<td>Minor version number</td>
</tr>
<tr>
<td>SIS_RevLetter</td>
<td>+17</td>
<td>1</td>
<td>B</td>
<td>Revision letter</td>
</tr>
<tr>
<td>SIS_CurScrGrp</td>
<td>+18</td>
<td>1</td>
<td>B</td>
<td>Fgnd screen group #</td>
</tr>
<tr>
<td>SIS_MaxScrGrp</td>
<td>+19</td>
<td>1</td>
<td>B</td>
<td>Maximum number of screen groups</td>
</tr>
<tr>
<td>SIS_HugeShfCnt</td>
<td>+1a</td>
<td>1</td>
<td>B</td>
<td>Shift count for huge segments</td>
</tr>
<tr>
<td>SIS_ProtMdOnly</td>
<td>+1b</td>
<td>1</td>
<td>B</td>
<td>Protect-mode-only indicator</td>
</tr>
<tr>
<td>SIS_FgndPID</td>
<td>+1c</td>
<td>2</td>
<td>W</td>
<td>Foreground process ID</td>
</tr>
<tr>
<td>SIS_Dynamic</td>
<td>+1e</td>
<td>1</td>
<td>B</td>
<td>Dynamic variation flag (1=enabled)</td>
</tr>
<tr>
<td>SIS_MxWait</td>
<td>+1f</td>
<td>1</td>
<td>B</td>
<td>Maxwait (seconds)</td>
</tr>
<tr>
<td>SIS_MnSlice</td>
<td>+20</td>
<td>2</td>
<td>W</td>
<td>Minimum timeslice (milliseconds)</td>
</tr>
</tbody>
</table>
SIS_MaxSlice  +22  2   W  Maximum timeslice (milliseconds)
SIS_BootDrv   +24  2   W  Drive from which system was booted
SIS_mec_table +26  20  B  Table of RAS Major Event Codes (MECs)
SIS_MaxVioWinSG +46  1   B  Max. no. of VIO windowable SG's
SIS_MaxPresMgrSG +47  1   B  Max. no. of Presentation Manager SG's
SIS_SYSLOG    +48  2   W  Error Logging Status
SIS_MMIOBase   +4a  2   W  Memory mapped I/O selector
SIS_MMIOAddr   +4c  4   D  Memory mapped I/O address
SIS_MaxVDMs    +50  1   B  Max. no. of Virtual DOS machines
SIS_SysLog     +51  1   B

SIS_SYSLOG flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF_LOGENABLE</td>
<td>0x0001</td>
<td>Logging enabled</td>
</tr>
<tr>
<td>LF_LOGAVAILABLE</td>
<td>0x0002</td>
<td>Logging available</td>
</tr>
</tbody>
</table>

--------------------------------------------

Process Information Block

Pointers
PTDA field ptda_avatib points to the PIB for the related process.
PIB field pib_pchenv points to the process' environment strings.

Locations
Allocated in the Process' Private Arena.

VM Owner
PIB owner id: ttkib (0xff3f) (also used for TIB ownership).
Environment Owner id: tkenv (0xff3e).

Format
PIB Process Information Block.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pib_ulpid</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Process I.D.</td>
</tr>
<tr>
<td>pib_uipid</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Parent process I.D.</td>
</tr>
<tr>
<td>pib_hmte</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Program (.EXE) module handle</td>
</tr>
<tr>
<td>pib_pchcmd</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Command line pointer</td>
</tr>
<tr>
<td>pib_pchenv</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Environment pointer</td>
</tr>
<tr>
<td>pib_flstatus</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Process' status bits</td>
</tr>
<tr>
<td>pib_ultype</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Process' type code</td>
</tr>
</tbody>
</table>
**pib_flstatus** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS_XITLST</td>
<td>0x01</td>
<td>Doing ExitList Processing</td>
</tr>
<tr>
<td>PS_XITTH1</td>
<td>0x02</td>
<td>Exiting thread 1</td>
</tr>
<tr>
<td>PS_XITALL</td>
<td>0x04</td>
<td>The whole process is exiting</td>
</tr>
<tr>
<td>PS_SYNC PARENT</td>
<td>0x10</td>
<td>Parent cares about termination</td>
</tr>
<tr>
<td>PS_WAIT PARENT</td>
<td>0x20</td>
<td>Parent did an exec-and-wait</td>
</tr>
<tr>
<td>PS_DYING</td>
<td>0x40</td>
<td>Process is dying</td>
</tr>
<tr>
<td>PS_EMBRYO</td>
<td>0x80</td>
<td>Process in embryonic state</td>
</tr>
</tbody>
</table>

**pib_utype** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIS_PT_FULLSCRN</td>
<td>0</td>
<td>Full screen app.</td>
</tr>
<tr>
<td>LIS_PT_REALMODE</td>
<td>1</td>
<td>Real mode process</td>
</tr>
<tr>
<td>PT_VDM</td>
<td>1</td>
<td>VDM</td>
</tr>
<tr>
<td>LIS_PT_VIOWIN</td>
<td>2</td>
<td>VIO windowable app.</td>
</tr>
<tr>
<td>LIS_PT_PRESMANGR</td>
<td>3</td>
<td>Presentation Manager app.</td>
</tr>
<tr>
<td>LIS_PT_DETACHED</td>
<td>4</td>
<td>Detached app.</td>
</tr>
</tbody>
</table>

-----------------------------

Thread Information Block

**Pointers**

TCB field **TCBptlib** points to the TIB for the related thread.

TIB field **tib_ptlib2** points to the associated TIB2.

GDT Selector 150b maps the TIB and is the default value for the FS register.

**Locations**

Allocated in the Process' Private Arena.

**VM Owner**

**tktib (0xff3f)** (also used for PIB ownership).

**Format**

TIB Thread Information Block system independent section.

**Field Name**  Offset Length Type Description

| tib_pexchain   | +0 | 4  | D  | Head of exception handler chain |
### TIB2: Thread Information Block

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tib2_ultid</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Thread I.D.</td>
</tr>
<tr>
<td>tib2_ulpri</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Thread priority</td>
</tr>
<tr>
<td>tib2_version</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Version number for this structure</td>
</tr>
<tr>
<td>tib2_usMCCount</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Must Complete count</td>
</tr>
<tr>
<td>tib2_fMCForceFlag</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>Must Complete force flag</td>
</tr>
</tbody>
</table>

---

### System Stack Frames Client Register Information

**Pointers**
- TCB field `TCB_pcriFrameType` points to the CRI.

**Locations**
- `_criISF` locates the Interrupt Stack Frame CRI.
- `_criTSF` locates the Trap Stack Frame CRI.
- `_criVSF` locates the VDM Stack Frame CRI.
- `_criSEF` locates the System Entry Stack Frame CRI.
- `_criPASCALSEF` locates the PASCAL System Entry Stack Frame CRI.
- `_criSSF` locates the SCI Stack Frame CRI.
- `_criDHF` locates the Device Help Stack Frame CRI.

**fpoldstack** contains a 32-bit far pointer to the ISF built by the Interrupt Router at interrupt time.

**VM Owner**
- `os2krnl (0xffaa)`

**Format**

### CRI: Client Register Information

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cri_ulSize</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>size of stack frame</td>
</tr>
<tr>
<td>cri_eax</td>
<td>+4</td>
<td>4</td>
<td>S</td>
<td>eax rip</td>
</tr>
<tr>
<td>cri_ebx</td>
<td>+8</td>
<td>4</td>
<td>S</td>
<td>ebx rip</td>
</tr>
</tbody>
</table>
RIP Register Information Packet

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rip_flags</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Flags</td>
</tr>
<tr>
<td>rip_offset</td>
<td>+2</td>
<td>4</td>
<td>W</td>
<td>Offset of register into stack frame</td>
</tr>
</tbody>
</table>

rip_flags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KM_RIP_INVALID</td>
<td>0x0001</td>
<td>invalid register</td>
</tr>
<tr>
<td>KM_RIP_INVALID_SET</td>
<td>0x0002</td>
<td>invalid register to set</td>
</tr>
<tr>
<td>KM_RIP_WORD</td>
<td>0x0004</td>
<td>word register</td>
</tr>
<tr>
<td>KM_RIP_TSD_RELATIVE</td>
<td>0x0008</td>
<td>rip_offset relative to TSD beginning</td>
</tr>
<tr>
<td>KM_RIP_C32</td>
<td>0x0010</td>
<td>32-bit C style call</td>
</tr>
</tbody>
</table>

ISF Interrupt Manager Stack Frame

This is what the stack frame looks like when the system is entered thru the interrupt manager during a hardware interrupt. For a hardware interrupt in a VDM context, the stack frame always needs to be a "VSF" type so the stack frame base is adjusted by ISF_VSF_START. The points the stack frame base to "isf.edi" in the regular interrupt frame. The interrupt stack frame has also been padded (ISF_STACK_PAD) between the general registers (EDI to EAX) and the hardware pushed registers (EIP to SS) with a dummy trap number and error code to look like the VSF stack frame.
Field Name          Offset Length Type Description
isf_CurrIntLevel   +0     4     D
isf_gs             +4     2     W
isf_padgs          +6     2     W
isf_fs             +8     2     W
isf_padfs          +a     2     W
isf_es             +c     2     W
isf_pades           +e     2     W
isf_ds             +10    2     W
isf_padds          +12    2     W
isf_edi             +14    4     D  start of VDM stack frame
isf esi             +18    4     D
isf ebp             +1c    4     D
isf_padesp         +20    4     D
isf_ebx             +24    4     D
isf edx             +28    4     D
isf ecx             +2c    4     D
isf eax             +30    4     D
isf_pad              +34    8     B
isf_eip             +3c    4     D
isf cs              +40    2     W
isf_padcs           +42    2     W
isf_eflag           +44    4     D
isf esp             +48    4     D
isf ss              +4c    2     W
isf_padss           +4e    2     W

TSF Trap or Exception Stack Frame
This is what the stack frame looks like when the system is entered thru a 386 exception (from protected mode).

Field Name          Offset Length Type Description
tsf_edi             +0     4     D
tsf esi             +4     4     D
tsf ebp             +8     4     D
tsf_padesp         +c     4     D
tsf_ebx             +10    4     D
tsf edx             +14    4     D
tsf ecx             +18    4     D
tsf eax             +1c    4     D
tsf gs              +20    2     W
KSF Kernel Stack Frame

This is what the stack frame looks like when the system is re-entered from ring 0. This is frame used for handling exception while already in kernel mode.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ksf_edi</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_esi</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_ebp</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_padesp</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_ebx</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_edx</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_ecx</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_eax</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_gs</td>
<td>+20</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_padgs</td>
<td>+22</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_fs</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_padfs</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_es</td>
<td>+28</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_pades</td>
<td>+2a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_ds</td>
<td>+2c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_padds</td>
<td>+2e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ksf_trapnum</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ksf_errcode</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
VSD VDM Process Stack Frame

This is what the stack frame looks like when the system is entered from a VDM thru a exception, software or hardware interrupt. Most of the 8086 emulation code uses this stack frame directly for performance. For hardware interrupts taken in a VDM (in either V86 mode or protected mode), the interrupt stack frame (see ISF) is adjusted to look like this frame.

The alternate stack frame holds the real or protected mode sensitive registers for the other mode. So when the VDM is in protected mode, the last V86 mode segment registers and CS:EIP, SS:ESP can be accessed, etc. Two things happen with we mode switch: 1) the alternate register set is exchanged with the regular set (vsf_eip to vsf_padgs is the exchanged with vsf_alteip to vsf_altpadgs), 2) the TSS's ESP0 value is changed to the appropriate place in the VSF structure. For V86 mode, ESP0 points to the beginning of the segment registers (vsf_gs/vsf_padgs) and for protected mode ESP0 points to the SS register (vsf_ss/vsf_padss). For protected mode entry, the segments registers are stored in vsf_ds to vsf_gs explicitly. This makes the V86 mode and protected mode stack frames the same for VDDs and the MVDM kernel code.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsf_edi</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_esi</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_ebp</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_padesp</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_ebx</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_edx</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_ecx</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_eax</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_trapnum</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_errcode</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_eip</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_cs</td>
<td>+2c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_padcs</td>
<td>+2e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_eflag</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_esp</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>vsf_ss</td>
<td>+38</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_padss</td>
<td>+3a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_es</td>
<td>+3c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_pades</td>
<td>+3e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_ds</td>
<td>+40</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_padds</td>
<td>+42</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_fs</td>
<td>+44</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_padfs</td>
<td>+46</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_gs</td>
<td>+48</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>vsf_padgs</td>
<td>+4a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
**SEF System Entry Stack Frame**

This is the frame put on the by the call gate system entry function (KMEnterKmodeCallGate or KMEnterKmodeAPI32).

This frame is used for:

- 32-bit C APIs, with C callable workers (criSEF)
- 16-bit PASCAL APIs, with C callable workers (criPASCALSEF)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sef_edi</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_esi</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_ebp</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_padesp</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sefxEBx</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_Edx</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_Ecx</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_Eax</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>sef_Gs</td>
<td>+20</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_padgs</td>
<td>+22</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_FS</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_PADFS</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_ES</td>
<td>+28</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_Pades</td>
<td>+2a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_DS</td>
<td>+2c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>sef_PADDS</td>
<td>+2e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
SCI System Call Interpreter Call Gate Stack Frame

This is what the stack frame looks like when the system is entered thru SCI via call gate using the KMEnterKmodeSCI function.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssf_edi</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_esi</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_ebp</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_padesp</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssfxEBx</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_edx</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_ecx</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_eax</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_gs</td>
<td>+20</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_padgs</td>
<td>+22</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_fs</td>
<td>+24</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_padfs</td>
<td>+26</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_es</td>
<td>+28</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_pades</td>
<td>+2a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_ds</td>
<td>+2c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_paddd</td>
<td>+2e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_thopadr</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_cbargs</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>The Most Significant Bit of cbargs in an SCI stack frame is used to denote that a 16 bit callgate is being used with the SCI mechanism. Used by Dynamic API's.</td>
</tr>
<tr>
<td>ssf_sciret</td>
<td>+38</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_eflag</td>
<td>+3a</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_eip</td>
<td>+3e</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>ssf_cs</td>
<td>+42</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>ssf_padcs</td>
<td>+42</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

DHF Device Help Stack Frame

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
TF Hardware Exception Trap Stack Frame

Stack frame for the trap manager before we go into kernel mode.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tf_trapnum</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>tf_errcode</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>tf_eip</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>tf_cs</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>tf_padcs</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>tf_eflags</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>tf_esp</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>tf_ss</td>
<td>+18</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>tf_padss</td>
<td>+1a</td>
<td>2</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

--------------------------------------------

Exit List Entry Data Structure
Points

PTDA field `ptda_pexitlist` points to the head of Exit List chain.

Locations

Allocated from the kernel heaps.

VM Owner

`tkextlsta (0xffc7)`

Format

**EXENT**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exl_next</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>link to next block/order</td>
</tr>
<tr>
<td>exl_addr</td>
<td>4</td>
<td>4</td>
<td>D</td>
<td>Exit list routine address</td>
</tr>
<tr>
<td>exl_class</td>
<td>8</td>
<td>2</td>
<td>W</td>
<td>order &amp; position 0 thru 0x1FF</td>
</tr>
<tr>
<td>exl_type</td>
<td>a</td>
<td>2</td>
<td>W</td>
<td>16:16 or 0:32</td>
</tr>
</tbody>
</table>

exl_type values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK_TYPE16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TK_TYPE32</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TK_TYPEDT</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Exception Handler Structures

Pointers

TIB field `tib_pexitchain` points to the head of the chain of EXCEPTIONREGISTRATIONRECORDs.

The 1st parameter to the exception handler points to the EXCEPTIONREPORTRECORD.

The 2nd parameter to the exception handler points to the EXCEPTIONREGISTRATIONRECORD.

The 3nd parameter to the exception handler points to the CONTEXTRECORD.

Locations

Allocated in the Process' Private Arena.

VM Owner

`tktib (0xff3f)` (also used for PIB ownership).

Format

**CONTEXTRECORD** Exception handler context record.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
ContextFlags +0 4 D Flags
          +4 6c S Floating point section
ctx_env +4 1c D Floating point environment
ctx_stack +20 50 S Floating point register stack (8 FPREG structures)
          +70 10 S Segment Register section
ctx_SegGs +70 4 D GS segment register
ctx_SegFs +74 4 D FS segment register
ctx_SegEs +78 4 D ES segment register
ctx_SegDs +7c 4 D DS segment register
          +80 18 S Integer Register section
ctx_RegEdi +80 4 D EDI register
ctx_RegEsi +84 4 D ESI register
ctx_RegEax +88 4 D EAX register
ctx_RegEbx +8c 4 D EBX register
ctx_RegEcx +90 4 D ECX register
ctx_RegEdx +94 4 D EDX register
          +98 18 S Control Register section
ctx_RegEbp +98 4 D EBP register
ctx_RegEip +9c 4 D EIP register
ctx_SegCs +a0 4 D CS selector
ctx_EFlags +a4 4 D Processor Flags register
ctx_RegEsp +a8 4 D ESP register
ctx_SegSs +ac 4 D SS segment register

FPREG Floating Point Register Stack Element

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>losig</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Low significance double-word</td>
</tr>
<tr>
<td>hisig</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>High significance double-word</td>
</tr>
<tr>
<td>signexp</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>Exponent</td>
</tr>
</tbody>
</table>

ContextFlags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT_CONTROL</td>
<td>0x00000001L</td>
<td>SS:ESP, CS:EIP, EFLAGS, EBP</td>
</tr>
</tbody>
</table>
CONTEXT_INTEGER  0x00000002L EAX, EBX, ECX, EDX, ESI, EDI
CONTEXT_SEGMENTS 0x00000004L DS, ES, FS, GS
CONTEXT_FLOATING_POINT 0x00000008L Numeric coprocessor state

**EXCEPTIONREPORTRECORD** Exception Handler Report Record.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExceptionNum</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Exception number</td>
</tr>
<tr>
<td>fHandlerFlags</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Exception attributes</td>
</tr>
<tr>
<td>NestedExceptionReportRecord</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Preceding exception’s report record if nested exception</td>
</tr>
<tr>
<td>ExceptionAddress</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Exception address</td>
</tr>
<tr>
<td>cParameters</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Size of exception specific information</td>
</tr>
<tr>
<td>ExceptionInfo</td>
<td>+14</td>
<td>10</td>
<td>D</td>
<td>Exception specific information</td>
</tr>
</tbody>
</table>

**fHandlerFlags** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EH_NONCONTINUABLE</td>
<td>0x1</td>
<td>Noncontinuable exception</td>
</tr>
<tr>
<td>EH_UNWINDING</td>
<td>0x2</td>
<td>Unwind is in progress</td>
</tr>
<tr>
<td>EH_EXIT_UNWIND</td>
<td>0x4</td>
<td>Exit unwind is in progress</td>
</tr>
<tr>
<td>EH_STACK_INVALID</td>
<td>0x8</td>
<td>Stack out of limits or unaligned</td>
</tr>
<tr>
<td>EH_NESTED_CALL</td>
<td>0x10</td>
<td>Nested exception handler call</td>
</tr>
</tbody>
</table>

**EXCEPTIONREGISTRATIONRECORD** Exception Handler Registration Record.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prev_structure</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Previously registered exception handler</td>
</tr>
<tr>
<td>ExceptionHandler</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Exception handler entry point address or -1 if end of chain</td>
</tr>
</tbody>
</table>

--------------------------------------------

Loader Control Block Reference

The following control blocks are described in this section:

- Module Table Entry (MTE)
- Swappable Module Table Entry (SMTE)
- Object Table Entry (OTE)
- Segment Table Entry (STE)
Module Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

For MTE formats for other versions of OS/2 see:

MTE for OS/2 V2.11

Pointers

_mte_h points to the head of the chain of MTEs.

_global_h points to head of the chain of library module MTEs.

Locations

Dynamically allocated from the kernel resident heap except for the two MTEs that represent kernel interfaces.

_DosMosMte locates the MTE in OS2KRNL for DOSCALLS.DLL.

_VDDModMte locates the MTE in OS2KRNL for MVDM.DLL.

VM Owner

Dynamically allocated MTEs have owner id Idrmte (0xffa6)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mte_flags2</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Module flags 2</td>
</tr>
<tr>
<td>mte_handle</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>the handle for this mte</td>
</tr>
<tr>
<td>mte_swapmte</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>link to swappable mte</td>
</tr>
<tr>
<td>mte_link</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>link to next mte</td>
</tr>
<tr>
<td>mte_flags1</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Module flags 1</td>
</tr>
<tr>
<td>mte_impmodcnt</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Num of entries in Imp Mod Name Tbl</td>
</tr>
<tr>
<td>mte_sfn</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>file system number for open file</td>
</tr>
<tr>
<td>mte_usecnt</td>
<td>+16</td>
<td>2</td>
<td>W</td>
<td>.EXE only - use count</td>
</tr>
<tr>
<td>mte_modname</td>
<td>+18</td>
<td>8</td>
<td>B</td>
<td>resident module name (zero extended)</td>
</tr>
</tbody>
</table>

**mte_flags1 flag definitions:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOAUTODS</td>
<td>0x00000000</td>
<td>No Auto DS exists</td>
</tr>
<tr>
<td>SOLO</td>
<td>0x00000001</td>
<td>Auto DS is shared</td>
</tr>
<tr>
<td>INSTANCEDS</td>
<td>0x00000002</td>
<td>Auto DS is not shared</td>
</tr>
<tr>
<td>INSTDLIBINIT</td>
<td>0x00000004</td>
<td>Per-instance Libinit</td>
</tr>
<tr>
<td>GINISETP</td>
<td>0x00000008</td>
<td>Global Init has been setup</td>
</tr>
<tr>
<td>NOINTERNFIXUPS</td>
<td>0x00000010</td>
<td>internal fixups in .EXE-.DLL applied</td>
</tr>
<tr>
<td>NOEXTERNFIXUPS</td>
<td>0x00000020</td>
<td>external fixups in .EXE-.DLL applied</td>
</tr>
<tr>
<td>CLASS_PROGRAM</td>
<td>0x00000040</td>
<td>Program class</td>
</tr>
<tr>
<td>Name</td>
<td>Bit Mask</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>MTE_FORMATMASK</td>
<td>0x0003</td>
<td>Module format mask</td>
</tr>
<tr>
<td>MTE_FORMATR1</td>
<td>0x0000</td>
<td>Module format reserved</td>
</tr>
<tr>
<td>MTE_FORMATNE</td>
<td>0x0001</td>
<td>Module format NE</td>
</tr>
<tr>
<td>MTE_FORMATLX</td>
<td>0x0002</td>
<td>Module format LX</td>
</tr>
<tr>
<td>MTE_SYSTEMDLL</td>
<td>0x0004</td>
<td>DLL exists in system list</td>
</tr>
</tbody>
</table>

---

**mte_flags2** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTE_MEDIATE</td>
<td>0x0000001000</td>
<td>File Media permits discarding</td>
</tr>
<tr>
<td>LDR_INVALID</td>
<td>0x0000002000</td>
<td>Module not loadable</td>
</tr>
<tr>
<td>PROGRAMMOD</td>
<td>0x00000000000</td>
<td>Program module</td>
</tr>
<tr>
<td>DEVDRVMOD</td>
<td>0x0000004000</td>
<td>Device driver module</td>
</tr>
<tr>
<td>LIBRARYMOD</td>
<td>0x0000008000</td>
<td>DLL module</td>
</tr>
<tr>
<td>VDDMOD</td>
<td>0x00000100000</td>
<td>VDD module</td>
</tr>
<tr>
<td>MVDDMOD</td>
<td>0x00000200000</td>
<td>Set if VDD Helper MTE (MVDM.DLL)</td>
</tr>
<tr>
<td>INGRAPH</td>
<td>0x00000400000</td>
<td>In Module Graph - see ldrgc.c</td>
</tr>
<tr>
<td>GINIDONE</td>
<td>0x00000800000</td>
<td>Global Init has finished</td>
</tr>
<tr>
<td>MTE_ALLOCATED</td>
<td>0x00001000000</td>
<td>Allocate specific or not</td>
</tr>
<tr>
<td>FSDMOD</td>
<td>0x00002000000</td>
<td>FSD MTE</td>
</tr>
<tr>
<td>FSHMOD</td>
<td>0x00004000000</td>
<td>FS helper MTE</td>
</tr>
<tr>
<td>MTE_LONGNAMES</td>
<td>0x00008000000</td>
<td>Module supports long-names</td>
</tr>
<tr>
<td>MTE_MEDIA</td>
<td>0x00010000000</td>
<td>File Media contiguous memory req</td>
</tr>
<tr>
<td>MTE_MEDIA16M</td>
<td>0x00020000000</td>
<td>File Media requires mem below 16M</td>
</tr>
<tr>
<td>MTSWAPLOAD</td>
<td>0x00040000000</td>
<td>Make code pages swap on load</td>
</tr>
<tr>
<td>MTE_PORTHOLE</td>
<td>0x00080000000</td>
<td>Port hole module</td>
</tr>
<tr>
<td>MTE_MEDIUM</td>
<td>0x01000000000</td>
<td>Module has shared memory protected</td>
</tr>
<tr>
<td>MTE_NEWMOD</td>
<td>0x02000000000</td>
<td>Newly added module</td>
</tr>
<tr>
<td>MTE_LLERİ</td>
<td>0x04000000000</td>
<td>Gets instance termination</td>
</tr>
<tr>
<td>MTE_SYMLOAD</td>
<td>0x08000000000</td>
<td>Set if debugger symbols loaded</td>
</tr>
</tbody>
</table>
Module Table Entry for OS/2 V2.11

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mte_flags2</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Module flags 2</td>
</tr>
<tr>
<td>mte_handle</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>the handle for this mte</td>
</tr>
<tr>
<td>mte_swapmte</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>link to swappable mte</td>
</tr>
<tr>
<td>mte_modname</td>
<td>+8</td>
<td>4</td>
<td>A</td>
<td>resident module name (zero extended)</td>
</tr>
<tr>
<td>mte_link</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>link to next mte</td>
</tr>
<tr>
<td>mte_flags1</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Module flags 1</td>
</tr>
<tr>
<td>mte_impmodcnt</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>Num of entries in Imp Mod Name Tbl</td>
</tr>
<tr>
<td>mte_sfn</td>
<td>+18</td>
<td>2</td>
<td>W</td>
<td>file system number for open file</td>
</tr>
<tr>
<td>mte_usecnt</td>
<td>+la</td>
<td>2</td>
<td>W</td>
<td>.EXE only - use count</td>
</tr>
</tbody>
</table>

Swappable Module Table Entry for OS/2 Warp V4.0

For SMTE formats for other versions of OS/2 see:

SMTE for OS/2 Warp V3.0

Pointers
MTE field mte_swapmte points to the associated SMTE.

Locations
Dynamically allocated from the kernel swappable heap except for the SMTE associated with DOSCALLS.DLL.
_DosMosMteSwappable locates the SMTE in OS2KRNL associated with DOSCALLS.DLL.
VM Owner

Dynamically allocated SMTEs have owner id ldrmte (0xffa6)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smte_mp_mpages</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Module # pages</td>
</tr>
<tr>
<td>smte_startobj</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Object # for instruction pointer</td>
</tr>
<tr>
<td>smte_eip</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>Extended instruction pointer</td>
</tr>
<tr>
<td>smte_stackobj</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>Object # for stack pointer</td>
</tr>
<tr>
<td>smte_esp</td>
<td>+10</td>
<td>4</td>
<td>D</td>
<td>Extended stack pointer</td>
</tr>
<tr>
<td>smte_pageshift</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>smte_fixupsize</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Fixup section size</td>
</tr>
<tr>
<td>smte_objtab</td>
<td>+1c</td>
<td>4</td>
<td>D</td>
<td>Object table offset</td>
</tr>
<tr>
<td>smte_obj_cnt</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>Number of objects in module</td>
</tr>
<tr>
<td>smte_objmap</td>
<td>+24</td>
<td>4</td>
<td>D</td>
<td>Object page map offset</td>
</tr>
<tr>
<td>smte_itermap</td>
<td>+28</td>
<td>4</td>
<td>D</td>
<td>Object iterated data map offset</td>
</tr>
<tr>
<td>smte_rsrc_tab</td>
<td>+2c</td>
<td>4</td>
<td>D</td>
<td>Offset of Resource Table</td>
</tr>
<tr>
<td>smte_rsrc_cnt</td>
<td>+30</td>
<td>4</td>
<td>D</td>
<td>Number of resource entries</td>
</tr>
<tr>
<td>smte_restab</td>
<td>+34</td>
<td>4</td>
<td>D</td>
<td>Offset of resident name table</td>
</tr>
<tr>
<td>smte_ent_tab</td>
<td>+38</td>
<td>4</td>
<td>D</td>
<td>Offset of Entry Table</td>
</tr>
<tr>
<td>smte_fp_agetab</td>
<td>+3c</td>
<td>4</td>
<td>D</td>
<td>Offset of Fixup Page Table</td>
</tr>
<tr>
<td>smte_frectab</td>
<td>+40</td>
<td>4</td>
<td>D</td>
<td>Offset of Fixup Record Table</td>
</tr>
<tr>
<td>smte_imp_mod</td>
<td>+44</td>
<td>4</td>
<td>D</td>
<td>Offset of Import Module Name Table</td>
</tr>
<tr>
<td>smte_imp_proc</td>
<td>+48</td>
<td>4</td>
<td>D</td>
<td>Offset of Imp Procedure Name Tab</td>
</tr>
<tr>
<td>smte_data_page</td>
<td>+4c</td>
<td>4</td>
<td>D</td>
<td>Offset of Enumerated Data Pages</td>
</tr>
<tr>
<td>smte_nrestab</td>
<td>+50</td>
<td>4</td>
<td>D</td>
<td>Offset of Non-resident Names Table</td>
</tr>
<tr>
<td>smte_cbn_restab</td>
<td>+54</td>
<td>4</td>
<td>D</td>
<td>Size of Non-resident Name Table</td>
</tr>
<tr>
<td>smte_autods</td>
<td>+58</td>
<td>4</td>
<td>D</td>
<td>Object # for automatic data object</td>
</tr>
<tr>
<td>smte_debug_info</td>
<td>+5c</td>
<td>4</td>
<td>D</td>
<td>Offset of the debugging info</td>
</tr>
<tr>
<td>smte_debug_len</td>
<td>+60</td>
<td>4</td>
<td>D</td>
<td>The len of the debug info in bytes</td>
</tr>
<tr>
<td>smte_heap_size</td>
<td>+64</td>
<td>4</td>
<td>D</td>
<td>use for converted 16-bit modules</td>
</tr>
<tr>
<td>smte_path</td>
<td>+68</td>
<td>4</td>
<td>D</td>
<td>full pathname</td>
</tr>
<tr>
<td>smte__sem_count</td>
<td>+6c</td>
<td>2</td>
<td>W</td>
<td>Count of threads waiting on MTE semaphore. 0 =&gt; semaphore is free</td>
</tr>
<tr>
<td>smte__sem_owner</td>
<td>+6e</td>
<td>2</td>
<td>W</td>
<td>Slot number of the owner of MTE semaphore</td>
</tr>
<tr>
<td>smte_pfile_cache</td>
<td>+70</td>
<td>4</td>
<td>D</td>
<td>Pointer to file cache for Dos32CacheModule</td>
</tr>
<tr>
<td>smte_stack_size</td>
<td>+74</td>
<td>4</td>
<td>D</td>
<td>Thread 1 Stack size from the exe header</td>
</tr>
<tr>
<td>smte_align_shift</td>
<td>+78</td>
<td>2</td>
<td>W</td>
<td>use for converted 16-bit modules</td>
</tr>
<tr>
<td>smte_NE_exp_ver</td>
<td>+7a</td>
<td>2</td>
<td>W</td>
<td>expver from NE header</td>
</tr>
<tr>
<td>smte_path_len</td>
<td>+7c</td>
<td>2</td>
<td>W</td>
<td>length of full pathname</td>
</tr>
</tbody>
</table>
Swappable Module Table Entry for OS/2 Warp V3.0

Field Name      Offset Length Type Description
mte_flags2      +0     2      W      Module flags 2
mte_handle      +2     2      W      the handle for this mte
mte_swapmte     +4     4      D      link to swappable mte
mte_link        +8     4      D      link to next mte
mte_flags1      +c     4      D      Module flags 1
mte_imppmodcnt   +10    4      D      Num of entries in Imp Mod Name Tbl
mte_sfn         +14    2      W      file system number for open file
mte_usecnt      +16    2      W      .EXE only - use count
mte_modname     +18    8      B      resident module name (zero extended)

Object Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers
SMTE field smte_objtab points to the associated OTE for 32-bit modules.

Locations
Dynamically allocated from the kernel swappable heap except for the SMTE associated with DOSCALLS.DLL.
dcm_ote_start locates the OTE in OS2KRNL associated with DOSCALLS.DLL.

VM Owner
Dynamically allocated OTEs have owner id ldrmte (0xffa6) and share the same heap block as their SMTE.

Format
Field Name      Offset Length Type Description
ote_size        +0     4      D      Object virtual size
ote_base        +4     4      D      Object base virtual address
ote_flags       +8     4      D      Attribute flags
ote_page +c 4 D Object page map index
ote_mapsize +10 4 D Num of entries in obj page map
ote_resu +14 4 S

ote_flags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJREAD</td>
<td>0x00000001L</td>
<td>Readable Object</td>
</tr>
<tr>
<td>OBJWRITE</td>
<td>0x00000002L</td>
<td>Writeable Object</td>
</tr>
<tr>
<td>OBJEXEC</td>
<td>0x00000004L</td>
<td>Executable Object</td>
</tr>
<tr>
<td>OBJRSRC</td>
<td>0x00000008L</td>
<td>Resource Object</td>
</tr>
<tr>
<td>OBJDISCARD</td>
<td>0x00000010L</td>
<td>Object is Discardable</td>
</tr>
<tr>
<td>OBJSHARED</td>
<td>0x00000020L</td>
<td>Object is Shared</td>
</tr>
<tr>
<td>OBJPRELOAD</td>
<td>0x00000040L</td>
<td>Object has preload pages</td>
</tr>
<tr>
<td>OBJINVALID</td>
<td>0x00000080L</td>
<td>Object has invalid pages</td>
</tr>
<tr>
<td>OBJZEROFIL</td>
<td>0x00000100L</td>
<td>Object has zero-filled pages</td>
</tr>
<tr>
<td>OBJRESIDENT</td>
<td>0x00000200L</td>
<td>Object is resident</td>
</tr>
<tr>
<td>OBJALIAS16</td>
<td>0x000001000L</td>
<td>16</td>
</tr>
<tr>
<td>OBJBIGDEF</td>
<td>0x000002000L</td>
<td>Big/Default bit setting</td>
</tr>
<tr>
<td>OBJCONFORM</td>
<td>0x000004000L</td>
<td>Object is conforming for code</td>
</tr>
<tr>
<td>OBJIOPL</td>
<td>0x000008000L</td>
<td>Object I/O privilege level</td>
</tr>
<tr>
<td>OBJMADEPRIV</td>
<td>0x040000000L</td>
<td>Object is made private for debug (now obsolete)</td>
</tr>
<tr>
<td>OBJALLOC</td>
<td>0x800000000L</td>
<td>Object is allocated used by loader</td>
</tr>
</tbody>
</table>

Segment Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers
SMTE field smte_objtab points to the associated STE for 16-bit modules.

Locations
Dynamically allocated from the kernel swappable heap.

VM Owner
Dynamically allocated STEs have owner id ldrmt (0xffa6) and share the same heap block as their SMTE.

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ste_offset</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>file offset to segment data</td>
</tr>
</tbody>
</table>
ste_size     +2     2      W    file data size
ste_flags    +4     2      W    type and attribute flags
ste_minsiz   +6     2      W    minimum allocation size
ste_seghdl   +8     2      W    segment handle
ste_selector +a     2      W    segment selector
ste_fixups   +c     4      D    fixup record storage

ste_flags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STE_TYPE_MASK</td>
<td>0x0001</td>
<td>segment type field</td>
</tr>
<tr>
<td>STE_CODE</td>
<td>0x0000</td>
<td>code segment type</td>
</tr>
<tr>
<td>STE_DATA</td>
<td>0x0001</td>
<td>data segment type</td>
</tr>
<tr>
<td>STE_PACKED</td>
<td>0x0002</td>
<td>segment is packed</td>
</tr>
<tr>
<td>STE_SEMAPHORE</td>
<td>0x0004</td>
<td>segment semaphore</td>
</tr>
<tr>
<td>STE_ITERATED</td>
<td>0x0008</td>
<td>segment data is iterated</td>
</tr>
<tr>
<td>STE_WAITING</td>
<td>0x0010</td>
<td>segment is waiting on semaphore</td>
</tr>
<tr>
<td>STE_SHARED</td>
<td>0x0020</td>
<td>segment can be shared</td>
</tr>
<tr>
<td>STE_PRELOAD</td>
<td>0x0040</td>
<td>segment is preload</td>
</tr>
<tr>
<td>STE_ERONLY</td>
<td>0x0080</td>
<td>excute only if code segment read only if data segment</td>
</tr>
<tr>
<td>STE_RELOCMINFO</td>
<td>0x0100</td>
<td>set if segment has reloc records</td>
</tr>
<tr>
<td>STE_CONFORM</td>
<td>0x0200</td>
<td>segment is conforming</td>
</tr>
<tr>
<td>STE_RING_2</td>
<td>0x0800</td>
<td>ring 2 selector</td>
</tr>
<tr>
<td>STE_RING_3</td>
<td>0x0C00</td>
<td>ring 3 selector</td>
</tr>
<tr>
<td>STE_HUGE</td>
<td>0x1000</td>
<td>huge segment</td>
</tr>
<tr>
<td>STE_PAGEABLE</td>
<td>0x2000</td>
<td>just a page can be faulted in</td>
</tr>
<tr>
<td>STE_PRESENT</td>
<td>0x2000</td>
<td>packed segment already loaded</td>
</tr>
<tr>
<td>STE_SELALLOC</td>
<td>0x4000</td>
<td>used to indicate sel allocated</td>
</tr>
<tr>
<td>STE_GDTSEG</td>
<td>0x8000</td>
<td>used to indicate GTD sel alloc</td>
</tr>
</tbody>
</table>

--------------------------------------------

Loader Demand Load Data OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

PTDA field ptda_pldrdldHead points to chain of modules loaded by DosLoadModule for a given process.

_pldrdldHeadKernel points to the head of kernel reference list of LDRLDs.
Locations
Dynamically allocated from the kernel resident heap.

VM Owner
ldrld (0xffa4)

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldrldld_plrdldNext</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Pointer to next ldrldld</td>
</tr>
<tr>
<td>ldrldld_hmte</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>handle of loaded module</td>
</tr>
<tr>
<td>ldrldld_cRef</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Number of times loaded</td>
</tr>
</tbody>
</table>

File System Control Block Reference

The following control blocks are described in this section:

- File System Control Block Entry (FSC)
- System File Table Entry (SFT)
- Master File Table Entry (MFT)
- Record Lock Record (RLR)
- Volume Parameter Block (VPB)
- Drive Parameter Block (DPB)
- Current Directory Structure (CDS)
- File System Buffer (BUF)
- Named Pipe Structures
- Anonymous Pipe Structures

An overview of the File System Control Blocks follows:

File System Control Block Diagrams

The following diagrams illustrate the relationships between various file system control blocks:

- Open Files - Application to System
- Open Files - System View
- Open Device - System View
- Shared Files with Locked Ranges
- Anonymous and Named Pipes
Open Files - Application to System
Open Files - System View
Open Device - System View
Open Device - System View
Shared Files with Locked Ranges
Shared File with 2 Locked Ranges
Anonymous and Named Pipes
File System Control Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

SFT field sf_FSC points to the associated FSC_ENTRY.

VPB field vp_FSC points to the associated FSC_ENTRY.

CDS field cd_ownerFSC points to the associated FSC_ENTRY.

SAS field SAS_dd_FSC contains the selector for FSCSEG.

GDT_FSC locates the GDT descriptor for the FSCSEG.

Locations

Dynamically allocated from the kernel resident heap.

VM Owner

fsc (0xff95).

Format

FSCSEG

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fss_Limit</td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>Offset PAST last allocated byte</td>
</tr>
<tr>
<td>fss_ShutdownFlags</td>
<td>2</td>
<td>2</td>
<td>W</td>
<td>flags for shutdown</td>
</tr>
<tr>
<td>fss_SDWaitCount</td>
<td>4</td>
<td>2</td>
<td>W</td>
<td>number of processes pending before shutdown</td>
</tr>
<tr>
<td>fss_pad</td>
<td>6</td>
<td>2</td>
<td>W</td>
<td>shutdown can commence (DWORD align)</td>
</tr>
</tbody>
</table>

FS_ENTRY

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_ATTRIBUTE</td>
<td>0-4</td>
<td>4</td>
<td>D</td>
<td>-&gt; FSD attribute. (in FSD memory)</td>
</tr>
<tr>
<td>FS_NAME</td>
<td>4-8</td>
<td>4</td>
<td>D</td>
<td>-&gt; FSD name. (in FSD memory)</td>
</tr>
<tr>
<td>FS_ATTACH</td>
<td>8</td>
<td>4</td>
<td>D</td>
<td>DosQFsAttach, DosFsAttach</td>
</tr>
<tr>
<td>FS_CHDIR</td>
<td>C</td>
<td>4</td>
<td>D</td>
<td>DosChdir</td>
</tr>
<tr>
<td>FS_CHGFILEPTR</td>
<td>10</td>
<td>4</td>
<td>D</td>
<td>DosChgFilePtr</td>
</tr>
<tr>
<td>FS_CLOSE</td>
<td>14</td>
<td>4</td>
<td>D</td>
<td>DosClose</td>
</tr>
<tr>
<td>FS_COPY</td>
<td>18</td>
<td>4</td>
<td>D</td>
<td>DosCopy</td>
</tr>
<tr>
<td>FS_DELETE</td>
<td>1C</td>
<td>4</td>
<td>D</td>
<td>DosDelete</td>
</tr>
<tr>
<td>FS_EXIT</td>
<td>20</td>
<td>4</td>
<td>D</td>
<td>DosExit</td>
</tr>
<tr>
<td>FS_FILEATTRIBUTE</td>
<td>24</td>
<td>4</td>
<td>D</td>
<td>DosFileInfo, DosSetFileMode</td>
</tr>
<tr>
<td>FS_FILEINFO</td>
<td>28</td>
<td>4</td>
<td>D</td>
<td>DosQFileInfo, DosSetFileInfo</td>
</tr>
<tr>
<td>FS_FILEIO</td>
<td>2C</td>
<td>4</td>
<td>D</td>
<td>DosFileIO</td>
</tr>
<tr>
<td>Symbol</td>
<td>Usage</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDCLOSE</td>
<td>30</td>
<td>DosFindClose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDFIRST</td>
<td>34</td>
<td>DosFindFirst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDFROMNAME</td>
<td>38</td>
<td>DosFindFromName-Private to server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDNEXT</td>
<td>3C</td>
<td>DosFindNext</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDNOTIFYCLOSE</td>
<td>40</td>
<td>DosFindNotifyClose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDNOTIFYFIRST</td>
<td>44</td>
<td>DosFindNotifyFirst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FINDNOTIFYNEXT</td>
<td>48</td>
<td>DosFindNotifyNext</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FSINFO</td>
<td>4C</td>
<td>DosQFsInfo, DosSetFsInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_INIT</td>
<td>50</td>
<td>-- No corresponding API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_IOCTL</td>
<td>54</td>
<td>DosDevIoctl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_MKDIR</td>
<td>58</td>
<td>DosMkdir</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_MOUNT</td>
<td>5C</td>
<td>-- No corresponding API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_MOVE</td>
<td>60</td>
<td>DosMove</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_NEWSIZE</td>
<td>64</td>
<td>DosNewsize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_NMPIPE</td>
<td>68</td>
<td>All named pipe related API's</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_OPENCREATE</td>
<td>6C</td>
<td>DosOpen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_PATHINFO</td>
<td>70</td>
<td>DosPathInfo, DosSetPathInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_PROCESSNAME</td>
<td>74</td>
<td>-- No corresponding API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_READ</td>
<td>78</td>
<td>DosRead, DosReadAsync</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_RMDIR</td>
<td>7C</td>
<td>DosRmdir</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SETSWAP</td>
<td>80</td>
<td>-- No corresponding API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_WRITE</td>
<td>84</td>
<td>DosWrite, DosWriteAsync</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_OPENPAGEFILE</td>
<td>88</td>
<td>init time only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_ALLOCATEPAGESPACE</td>
<td>8C</td>
<td>size swap file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_CANCELLLOCKREQUEST</td>
<td>90</td>
<td>DosCancelLockRequest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FILELOCKS</td>
<td>94</td>
<td>DosSetFileLocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_VERIFYUNCNAME</td>
<td>98</td>
<td>Used to save function addresses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_COMMIT</td>
<td>9C</td>
<td>DosBufReset, DosClose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_DOPAGEIO</td>
<td>A0</td>
<td>perform paging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FSCtl</td>
<td>A4</td>
<td>DosFsCtl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_FLUSHBUF</td>
<td>A8</td>
<td>DosBufReset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SHUTDOWN</td>
<td>AC</td>
<td>DosShutdown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SDCHGFILEPTR</td>
<td>B0</td>
<td>Used to save function addresses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SDFILESINFO</td>
<td>B4</td>
<td>at shutdown time. These functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SDREAD</td>
<td>B8</td>
<td>will only be called by shutdown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS_SDWRITE</td>
<td>BC</td>
<td>filters.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FS_ATTRIBUTE** flag definitions
### Name Bit Mask Description

- **FS_ATTR_REMOTE** 0x0001 0 = local FSD, 1 = remote FSD
- **FS_ATTR_UNC** 0x0002 0 = normal, 1 = this is UNC FSD
- **FS_ATTR_LOCKINFO** 0x0004 0 = no notice, 1 = notify filelocks
- **FS_ATTR_LVL7** 0x0008 0 = no level 7 requests, 1 = yes
- **FS_ATTR_PIPESVR** 0x0010 0 = don't FSD on PIPE req, 1 = yes
- **FS_ATTR_Verno** 0x7000 bits 28-30 version no
- **FS_ATTR_EA** 0x8000 bit 31 -> 1 = extended attribute

### FS_COMMIT flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_COMMIT_ALL</td>
<td>2</td>
<td>all handles commit</td>
</tr>
<tr>
<td>FS_COMMIT_ONE</td>
<td>1</td>
<td>one handle commit</td>
</tr>
</tbody>
</table>

### Equates for close type

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_CL_ORDINARY</td>
<td>0</td>
<td>ordinary close</td>
</tr>
<tr>
<td>FS_CL_FORPROC</td>
<td>1</td>
<td>final close for process</td>
</tr>
<tr>
<td>FS_CL_FORSYS</td>
<td>2</td>
<td>final close for system</td>
</tr>
</tbody>
</table>

### FSCNmEntstruc

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSCNmEnt_Emulation</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>FSCNmEnt_Group</td>
<td>4</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FSCNmEnt_NameLen</td>
<td>5</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>FSCNmEnt_ProcName</td>
<td>6</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

### MFS_ENTRY

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFS_CHGFILEPTR</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>DosChgFilePtr</td>
</tr>
<tr>
<td>MFS_CLOSE</td>
<td>4</td>
<td>4</td>
<td>D</td>
<td>DosClose</td>
</tr>
<tr>
<td>MFS_INIT</td>
<td>8</td>
<td>4</td>
<td>D</td>
<td>-- No corresponding API</td>
</tr>
<tr>
<td>MFS_OPEN</td>
<td>C</td>
<td>4</td>
<td>D</td>
<td>DosOpen</td>
</tr>
<tr>
<td>MFS_READ</td>
<td>10</td>
<td>4</td>
<td>D</td>
<td>DosRead, DosReadAsync</td>
</tr>
<tr>
<td>MFS_TERM</td>
<td>14</td>
<td>4</td>
<td>D</td>
<td>DosRead, DosReadAsync</td>
</tr>
</tbody>
</table>

### uncfscentrstruc
System File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0
Pointers

MFT field mft_sptr points to the associated sf_entry.

RLR field rlr_sptr points to the associated sf_entry.

SAS field SAS_file_SFT contains the selector for the SFT segment table.

NP field np_ssft points to the server SFT for a named pipe.

NP field np_csft points to the client SFT for a named pipe.

GDT_SFT points to the client SFT for a named pipe.

Locations

Dynamically allocated from the system arena.

VM Owner

sft (0xffa1).

Format

SFT

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sft_link</td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>selector for next chunk of table</td>
</tr>
<tr>
<td>sft_count</td>
<td>2</td>
<td>2</td>
<td>W</td>
<td>number of entries in this block</td>
</tr>
<tr>
<td>sft_handle</td>
<td>4</td>
<td>2</td>
<td>W</td>
<td>handle of segment holding this block</td>
</tr>
<tr>
<td>sft_inshutdown</td>
<td>6</td>
<td>2</td>
<td>W</td>
<td>flags for shutdown</td>
</tr>
</tbody>
</table>

sf_entry

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sf_ref_count</td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>number of processes sharing entry</td>
</tr>
<tr>
<td>sf_usercnt</td>
<td>2</td>
<td>2</td>
<td>W</td>
<td>For files: number of threads waiting for access to sf_entry. For devices: number of threads using this sf_entry.</td>
</tr>
<tr>
<td>reserved</td>
<td>4</td>
<td>1</td>
<td>B</td>
<td>Used to be attr of file - moved to * independent part of the SFT for general * access</td>
</tr>
<tr>
<td>sf_flags</td>
<td>5</td>
<td>2</td>
<td>W</td>
<td>Bits 8-15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 15 = 1 if remote file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0 if local file or device</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 14 = not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 13 = Pipe bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 12 = FCB bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1 if fcb sft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0 if normal sft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 11 = if Pipe,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0 if anonymous pipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1 if named pipe</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 10 == sf_inuse = sf_entry is in use by some thread, ie busy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 9 == sfWant = some thrd blocked waiting to use the sf_entry</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit 8 == sf_noJFN, no handle allocated for sft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bits 0-7 (old FCB_devid bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If remote file or local file, bit 6=0 if dirty Device ID number, bits 0-5 if local file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 7=0 for local file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=1 for local I/O device</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If local I/O device, bit 6=0 if EOF (input)</td>
</tr>
</tbody>
</table>
Bit 5=1 if Raw mode
Bit 0=1 if console input device
Bit 1=1 if console output device
Bit 2=1 if null device
Bit 3=1 if clock device

**sf** _flags2_ flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF_FORMAT_MOUNT</td>
<td>0x8000</td>
<td>True if a format mount was done, and still in effect</td>
</tr>
<tr>
<td>SF_BEGINFORMAT_FAILED</td>
<td>0x4000</td>
<td>True if a beginformat ioctl failed</td>
</tr>
<tr>
<td>SFF2_LDRBINARYSEM</td>
<td>0x2000</td>
<td>'ON' if SFT owned by some thread</td>
</tr>
<tr>
<td>SF_SRVRDR</td>
<td>0x1000</td>
<td>serving pipe redirection in effect</td>
</tr>
<tr>
<td>SFF2_LOCKED_DRIVE</td>
<td>0x0800</td>
<td>A LOCK was issued on this direct access handle to lock the drive.</td>
</tr>
<tr>
<td>SFF2_SPOOLED</td>
<td>0x0400</td>
<td>File is spooled</td>
</tr>
<tr>
<td>SFF2_DATAWRITTEN</td>
<td>0x0200</td>
<td>Data written to file</td>
</tr>
</tbody>
</table>
SFF2 Consistency     0x0180   consistency bits
SFF2_CANCELJOB       0x0040   spool job has been canceled*/ ;whs
SFF2_NONSPOOLED      0x0020   File is nonspooled; going to printer
SFF2_STPTHINFDN      0x0010   SetPathInfo done, don't set archive
sff2_RA_ON           0x0008   Readahead started
sff2_UNC             0x0004   UNC object
sff2_isfree          0x0002   this SFT is on free list (unused)
sff2_RA_BIG          0x0001   Big Readahead

sfdFATFS

Field Name Offset Length Type Description
sfdFAT_firFILEclus   0    2    W    First cluster of file (bit 15 = 0)
sfdFAT_cluspos       2    2    W    Position of last cluster accessed
sfdFAT_lstclus       4    2    W    Last cluster accessed
sfdFAT_dirsec        6    4    D    Sector # of directory sector for this file
sfdFAT_dirpos        A    1    B    Offset of this entry in the above
sfdFAT_EAHandle      B    2    W    starting cluster of EAs
sfdFAT_bRAReads      18   4    D    # of consecutive reads within range
sfdFAT_bRABigReads   1C   4    D    # of consecutive big reads
sfdFAT_fldMask       20   4    D    Unique File Dirty Mask
sfdFAT_pSFT          24   4    D    Linear address of SFT
sfdFAT_ulNextRA      28   4    D    Position where next rahead starts
sfdFAT_bBufRun       2C   4    D    Number of sectors in rahead run
sfdFAT_LastFATSec    30   2    W    last FAT sector added to chain

sftfsd

Field Name Offset Length Type Description
sfd_work[50]         0    32   B

sftfsi

Field Name Offset Length Type Description
sfi_mode            0    2    W    mode of access or high bit on if FCB
sfi_mode2           2    2    W    additional openmode bits for DosOpen2
sfi_hVPB            4    2    W    handle of volume
sfi_ctime           6    2    W    Creation time of file
<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STYPE_FILE</td>
<td>0x0000</td>
<td>file</td>
</tr>
<tr>
<td>STYPE_DEVICE</td>
<td>0x0001</td>
<td>device</td>
</tr>
<tr>
<td>STYPE_NMPipe</td>
<td>0x0002</td>
<td>named pipe</td>
</tr>
<tr>
<td>STYPE_FCB</td>
<td>0x0004</td>
<td>SFT is for an FCB</td>
</tr>
</tbody>
</table>

Master File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRICT kernel

For **MTE** formats for other versions of OS/2 see:

**MTE** for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

**Pointers**

- SFT field **sf_MFT** points to the associated MFT entry.
- SAS field **SAS_file_MFT** points to the PTREE head for the MFT PTREE.

**Locations**

- Dynamically allocated from the kernel resident heap.

**VM Owner**

- mft (0xff9e).

**Format**
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mft_ksem</td>
<td>+0</td>
<td>10</td>
<td>S</td>
<td>multi read/single write semaphore</td>
</tr>
<tr>
<td>mft_lptr</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>16 bit offset to first LOCK record</td>
</tr>
<tr>
<td>mft_sptr</td>
<td>+12</td>
<td>4</td>
<td>D</td>
<td>16 bit FAR pointer to first SFT in chain</td>
</tr>
<tr>
<td>mft_pCMap</td>
<td>+16</td>
<td>4</td>
<td>D</td>
<td>32 bit FLAT pointer to cluster map</td>
</tr>
<tr>
<td>mft_CMapKSem</td>
<td>+1a</td>
<td>10</td>
<td>S</td>
<td>semaphore for access to pCMap</td>
</tr>
<tr>
<td>mft_opflags</td>
<td>+2a</td>
<td>2</td>
<td>W</td>
<td>oplock flags</td>
</tr>
<tr>
<td>mft_serl</td>
<td>+2c</td>
<td>2</td>
<td>W</td>
<td>serial number for PCB checking</td>
</tr>
<tr>
<td>mft_flags</td>
<td>+2e</td>
<td>2</td>
<td>W</td>
<td>general purpose MFT flags</td>
</tr>
<tr>
<td>mft_signature</td>
<td>+30</td>
<td>2</td>
<td>W</td>
<td>for sanity check</td>
</tr>
<tr>
<td>mft_hvpb</td>
<td>+32</td>
<td>2</td>
<td>W</td>
<td>handle of vpb</td>
</tr>
<tr>
<td>mft_name</td>
<td>+34</td>
<td>1</td>
<td>B</td>
<td>start of name string (zero terminated)</td>
</tr>
</tbody>
</table>

**mft_flags** flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mft_pagerheap</td>
<td>0x0001</td>
<td>MFT is allocated on pager heap</td>
</tr>
<tr>
<td>MFT_DEFAULTHEAP</td>
<td>0x0</td>
<td>MFT is allocated on kernel (heap default MFT heap)</td>
</tr>
</tbody>
</table>

**mft_opflags** flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mft_opnolock</td>
<td>0</td>
<td>no oplock or opbatch on file</td>
</tr>
<tr>
<td>mft_oplock</td>
<td>1</td>
<td>oplock on file</td>
</tr>
<tr>
<td>mft_opbatch</td>
<td>2</td>
<td>opbatch on file</td>
</tr>
<tr>
<td>mft_opbreak</td>
<td>4</td>
<td>oplock/batch cleanup in process</td>
</tr>
<tr>
<td>mft_opbreakfailed</td>
<td>8</td>
<td>oplock/batch cleanup failed</td>
</tr>
</tbody>
</table>

--------------------------------------------

Master File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mft_ksem</td>
<td>+0</td>
<td>C</td>
<td>S</td>
<td>multi read/single write semaphore</td>
</tr>
<tr>
<td>mft_lptr</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>16 bit offset to first LOCK record</td>
</tr>
</tbody>
</table>
**Record Lock Record for OS/2 Warp V4.0 and OS/2 Warp V3.0**

**Pointers**

MFT field `mft_lptr` contains the offset within the RLR segment to the first RLR associated with the MFT.

**Locations**

Dynamically allocated from the system arena.

**VM Owner**

`fsreclok (0xff47)`.

**Format**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rlr_next</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>16 bit offset to next RLR. 0 if end</td>
</tr>
<tr>
<td>rlr_prev</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>16 bit offset to prev RLR. 0 if SFT</td>
</tr>
<tr>
<td>rlr_fba</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>offset of first byte of locked region</td>
</tr>
<tr>
<td>rlr_lba</td>
<td>+8</td>
<td>4</td>
<td>D</td>
<td>offset of last byte of locked region</td>
</tr>
<tr>
<td>rlr_sptr</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>16:16 FAR pointer to SFT</td>
</tr>
<tr>
<td>rlr_UID</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>lock issuer's user ID</td>
</tr>
<tr>
<td>rlr_PID</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>lock issuer's process ID</td>
</tr>
<tr>
<td>rlr_PDB</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>lock issuer's PDB, 0 for non-3xBox</td>
</tr>
<tr>
<td>rlr_flags</td>
<td>+16</td>
<td>1</td>
<td>B</td>
<td>flags</td>
</tr>
</tbody>
</table>

**rlr_flags** flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLR_EXCLUSIVE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RLR_SHARED</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Volume Parameter Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

- SFT field `sfi_hVPB` contains the offset within the VPB segment of the associated VPB.
- MFT field `mft_hVPB` contains the offset within the VPB segment of the associated VPB.
- DPB field `dpb_hVPB` contains the offset within the VPB segment of the associated VPB.
- CDS field `cdi_hVPB` contains the offset within the VPB segment of the associated VPB.
- `GDT_VPB` locates the GDT descriptor for the VPB segment.

Locations

- VPB segment is dynamically allocated from the kernel resident heap.

VM Owner

- `vpb (0xffa2)`.

Format

**vpb**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vpb_flink</code></td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>handle of forward link</td>
</tr>
<tr>
<td><code>vpb_blink</code></td>
<td>2</td>
<td>2</td>
<td>W</td>
<td>handle of back link</td>
</tr>
<tr>
<td><code>vpb_ref_count</code></td>
<td>4</td>
<td>2</td>
<td>W</td>
<td>count of objects that point to VPB</td>
</tr>
<tr>
<td><code>vpb_search_count</code></td>
<td>6</td>
<td>2</td>
<td>W</td>
<td>count of searches that point to VPB</td>
</tr>
<tr>
<td><code>vpb_first_access</code></td>
<td>8</td>
<td>1</td>
<td>B</td>
<td>initialized to -1 to force a media</td>
</tr>
<tr>
<td><code>vpb_signature</code></td>
<td>9</td>
<td>2</td>
<td>W</td>
<td>Signature specifying VPB validity</td>
</tr>
<tr>
<td><code>vpb_flags</code></td>
<td>B</td>
<td>1</td>
<td>B</td>
<td>flags (bits 7,6,3-0 defined below)</td>
</tr>
<tr>
<td><code>vpb_fMisc</code></td>
<td>C</td>
<td>1</td>
<td>B</td>
<td>More flags (bit 7 defined below)</td>
</tr>
<tr>
<td><code>vpb_FSC</code></td>
<td>D</td>
<td>4</td>
<td>D</td>
<td>Pointer to the file system control block (FSC).</td>
</tr>
<tr>
<td><code>vpb_fsd</code></td>
<td>11</td>
<td>40</td>
<td>S</td>
<td>File system dependent section</td>
</tr>
<tr>
<td><code>vpb_fsi</code></td>
<td>51</td>
<td>2C</td>
<td>S</td>
<td>File system independent section</td>
</tr>
</tbody>
</table>

**vpb_signature** values

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPB_VALID</td>
<td>0x444A</td>
<td></td>
</tr>
</tbody>
</table>
VPP_INVALID 0x4A47

**vpb_ID values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNREAD_ID1</td>
<td>0x4A52</td>
<td>Media unreadable</td>
</tr>
<tr>
<td>UNREAD_ID2</td>
<td>0x534E</td>
<td>Media unreadable</td>
</tr>
<tr>
<td>DAMAGED_ID1</td>
<td>0x0000</td>
<td>Media damaged but recognised by IFS</td>
</tr>
<tr>
<td>DAMAGED_ID2</td>
<td>0x0000</td>
<td>Media damaged but recognised by IFS</td>
</tr>
</tbody>
</table>

**vpb_falgs flag definitions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPBCHECK</td>
<td>0x01</td>
<td>a volume ID check is going on for this VPB</td>
</tr>
<tr>
<td>VPBNEWBOOT</td>
<td>0x02</td>
<td>new format disk</td>
</tr>
<tr>
<td>VPMOUNT</td>
<td>0x04</td>
<td>Mount in progress</td>
</tr>
<tr>
<td>VPBFORMATMOUNT</td>
<td>0x08</td>
<td>FormatMount done, not cleared</td>
</tr>
<tr>
<td>VPBINVALID</td>
<td>0x10</td>
<td>volume formatted - old vpb invalid</td>
</tr>
<tr>
<td>VPBINICACHE</td>
<td>0x20</td>
<td>Initializing Cache Data</td>
</tr>
<tr>
<td>VPBSETVID</td>
<td>0x40</td>
<td>vid set is in progress</td>
</tr>
<tr>
<td>VPBALLOCATE</td>
<td>0x80</td>
<td>cluster allocation in progress</td>
</tr>
</tbody>
</table>

**vpb_fMisc flag definitions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPB_FM_WRITEABLE</td>
<td>0x01</td>
<td>Set if we know volume can be written</td>
</tr>
<tr>
<td>VPB_FM_UNKNOWN</td>
<td>0x02</td>
<td>Set if no FATs and not claimed by FSD</td>
</tr>
<tr>
<td>VPB_REMOTE_DRIVE</td>
<td>0x04</td>
<td>set for attaches of remote drives</td>
</tr>
<tr>
<td>VPB_FM_ALLOCSHWAIT</td>
<td>0x08</td>
<td>Set if somebody wants alloc access so that they can get some disk clusters for this volume</td>
</tr>
<tr>
<td>VPB_FMALLOCEXWAIT</td>
<td>0x10</td>
<td>excl.access wait for somebody who wants to release some clusters</td>
</tr>
<tr>
<td>VPB_FM_INITCACHE_ERROR</td>
<td>0x20</td>
<td>Error initializing cache</td>
</tr>
<tr>
<td>VPB_FM_INITCACHE_DONE</td>
<td>0x40</td>
<td></td>
</tr>
</tbody>
</table>

**vpbfsd**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpd_work[64]</td>
<td>0</td>
<td>40</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>
### Field Names, Offsets, Lengths, and Types

**vpbfsi**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpi_ID</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>32 bit unique ID of file (See UNREAD_IDx, DAMAGED_IDx)</td>
</tr>
<tr>
<td>vpi_pDPB</td>
<td>4</td>
<td>4</td>
<td>D</td>
<td>Drive volume is in</td>
</tr>
<tr>
<td>vpi_cbSector</td>
<td>8</td>
<td>2</td>
<td>W</td>
<td>Size of physical sector in bytes</td>
</tr>
<tr>
<td>vpi_totsec</td>
<td>A</td>
<td>4</td>
<td>D</td>
<td>Total number of sectors on medium</td>
</tr>
<tr>
<td>vpi_trksec</td>
<td>E</td>
<td>2</td>
<td>W</td>
<td>Sectors per track on medium</td>
</tr>
<tr>
<td>vpi_nhead</td>
<td>10</td>
<td>2</td>
<td>W</td>
<td>Number of heads in device</td>
</tr>
<tr>
<td>vpi_text[12]</td>
<td>12</td>
<td>C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>vpi_pDCS</td>
<td>1E</td>
<td>4</td>
<td>D</td>
<td>device capability struc</td>
</tr>
<tr>
<td>vpi_pVCS</td>
<td>22</td>
<td>4</td>
<td>D</td>
<td>volume characteristic struc</td>
</tr>
<tr>
<td>vpi_drive</td>
<td>26</td>
<td>1</td>
<td>B</td>
<td>drive (0=A)</td>
</tr>
<tr>
<td>vpi_unit</td>
<td>27</td>
<td>1</td>
<td>B</td>
<td>unit</td>
</tr>
<tr>
<td>vpi_flags</td>
<td>28</td>
<td>2</td>
<td>W</td>
<td>flags for memory restrictions</td>
</tr>
</tbody>
</table>

**vpdFATFS**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpdFAT_cluster_mask</td>
<td>0</td>
<td>1</td>
<td>B</td>
<td>Sectors/cluster - 1</td>
</tr>
<tr>
<td>vpdFAT_cluster_shift</td>
<td>1</td>
<td>1</td>
<td>B</td>
<td>Log2 of sectors/cluster</td>
</tr>
<tr>
<td>vpdFAT_first_FAT</td>
<td>2</td>
<td>2</td>
<td>W</td>
<td>Starting record of FATs</td>
</tr>
<tr>
<td>vpdFAT_FAT_count</td>
<td>4</td>
<td>1</td>
<td>B</td>
<td>Number of FATs for this drive</td>
</tr>
<tr>
<td>vpdFAT_root_entries</td>
<td>5</td>
<td>2</td>
<td>W</td>
<td>Number of directory entries</td>
</tr>
<tr>
<td>vpdFAT_first_sector</td>
<td>7</td>
<td>4</td>
<td>D</td>
<td>First sector of first cluster</td>
</tr>
<tr>
<td>vpdFAT_max_cluster</td>
<td>B</td>
<td>2</td>
<td>W</td>
<td>Number of clusters on drive + 1</td>
</tr>
<tr>
<td>vpdFAT_FAT_size</td>
<td>D</td>
<td>2</td>
<td>W</td>
<td>Number of records occupied by FAT</td>
</tr>
<tr>
<td>vpdFAT_dir_sector</td>
<td>F</td>
<td>4</td>
<td>D</td>
<td>Starting record of directory</td>
</tr>
<tr>
<td>vpdFAT_media</td>
<td>13</td>
<td>1</td>
<td>B</td>
<td>Media byte (duplicate of VPB)</td>
</tr>
<tr>
<td>vpdFAT_next_free</td>
<td>14</td>
<td>2</td>
<td>W</td>
<td>Cluster # of last allocated cluster</td>
</tr>
<tr>
<td>vpdFAT_free_cnt</td>
<td>16</td>
<td>2</td>
<td>W</td>
<td>Count of free clusters, -1 if unknown</td>
</tr>
<tr>
<td>vpdFAT_FATentrysize</td>
<td>18</td>
<td>1</td>
<td>B</td>
<td>12 or 16 - can you guess why ??? @0</td>
</tr>
<tr>
<td>vpdFAT_IDsector</td>
<td>19</td>
<td>4</td>
<td>D</td>
<td>sector number of ID</td>
</tr>
<tr>
<td>vpdFAT_minEOF</td>
<td>1D</td>
<td>2</td>
<td>W</td>
<td>minimum EOF cluster value: 12-bit -&gt; PPB, 16-bit -&gt; PPPP</td>
</tr>
<tr>
<td>vpdFAT_access</td>
<td>1F</td>
<td>2</td>
<td>W</td>
<td>whether rmdir XOR mov dir XOR (chdir mkdir OR mov file OR create)* has access to volume</td>
</tr>
<tr>
<td>vpdFAT_accwait</td>
<td>21</td>
<td>2</td>
<td>W</td>
<td>who's waiting for access</td>
</tr>
<tr>
<td>vpdFAT_alloc</td>
<td>23</td>
<td>2</td>
<td>W</td>
<td>whether disk cluster alloc OR release</td>
</tr>
</tbody>
</table>
vpdFAT_eaflags  25  2  W   flags for EA usage
vpdFAT_eareaders  27  2  W   number of threads with pending reads
vpdFAT_eawaiters  29  2  W   number of threads waiting to run
vpdFAT_eahandles  2B  2  W   number of handles in EAOffTable
vpdFAT_pEASFT  2D  4  D   SFT for "EA DATA. SF"
vpdFAT_pBadSector  31  4  D   Ptr for Bad sectors data
vpdFAT_pClusBitMap  35  4  D   Ptr to free cluster bit map
vpdFAT_pNextFreeBitMap  39  4  D   Ptr to next free bit map position
vpdFAT_cNextFreeBitMap  3D  2  W   Count of dwords remaining in bit map

vpdFAT_eaflags flag definitions

Name           Bit Mask Description
eavpb_fileopen 0x0001   the EA file on this volume is open
eavpb_changing 0x0002   the EA file is changing
eavpb_dooropen 0x0004   the drive door has been opened

--------------------------------------------

Drive Parameter Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers
VPB field vpi_pDPB points to the associated DPB.

Locations
DPB segment is dynamically allocated from the kernel resident heap.

VM Owner
dpb (0xff96).

Format

DPB

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpb_drive</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>Logical drive # assoc with DPB</td>
</tr>
<tr>
<td>dpb_unit</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>Driver unit number of DPB</td>
</tr>
<tr>
<td>dpb_driver_addr</td>
<td>+2</td>
<td>4</td>
<td>D</td>
<td>Pointer to driver</td>
</tr>
<tr>
<td>dpb_next_dpb</td>
<td>+6</td>
<td>4</td>
<td>D</td>
<td>Pointer to next Drive parameter block</td>
</tr>
<tr>
<td>dpb_cbSector</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>sector size (for volume checking)</td>
</tr>
<tr>
<td>dpb_first_FAT</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>sector of 1st FAT (for ancient dev drivers)</td>
</tr>
<tr>
<td>dpb_toggle_time</td>
<td>+e</td>
<td>4</td>
<td>D</td>
<td>time of last drive toggle</td>
</tr>
</tbody>
</table>
**dpb** flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPBCHECK</td>
<td>0x10</td>
<td>disk in drive is being removed/checked for VPB</td>
</tr>
<tr>
<td>DPBNONREMOV</td>
<td>0x20</td>
<td>1 =&gt; drive supports non-removable media</td>
</tr>
<tr>
<td>DPBVCRAMDISK</td>
<td>0x40</td>
<td>Ram Disk Driver</td>
</tr>
</tbody>
</table>

**DPB3X**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpb3x_drive</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>Logical drive # assoc with DPB (A=0,B=1,...)</td>
</tr>
<tr>
<td>dpb3x_UNIT</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>Driver unit number of DPB</td>
</tr>
<tr>
<td>dpb3x_sector_size</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Size of physical sector in bytes</td>
</tr>
<tr>
<td>dpb3x_cluster_mask</td>
<td>+4</td>
<td>1</td>
<td>B</td>
<td>Sectors/cluster - 1</td>
</tr>
<tr>
<td>dpb3x_cluster_shift</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td>Log2 of sectors/cluster</td>
</tr>
<tr>
<td>dpb3x_first_FAT</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Starting record of FATs</td>
</tr>
<tr>
<td>dpb3x_FAT_count</td>
<td>+8</td>
<td>1</td>
<td>B</td>
<td>Number of FATs for this drive</td>
</tr>
<tr>
<td>dpb3x_root_entries</td>
<td>+9</td>
<td>2</td>
<td>W</td>
<td>Number of directory entries</td>
</tr>
<tr>
<td>dpb3x_first_sector</td>
<td>+b</td>
<td>2</td>
<td>W</td>
<td>First sector of first cluster</td>
</tr>
<tr>
<td>dpb3x_max_cluster</td>
<td>+d</td>
<td>2</td>
<td>W</td>
<td>Number of clusters on drive + 1</td>
</tr>
<tr>
<td>dpb3x_FAT_size</td>
<td>+f</td>
<td>1</td>
<td>B</td>
<td>Number of records occupied by FAT</td>
</tr>
<tr>
<td>dpb3x_dir_sector</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>Starting record of directory</td>
</tr>
<tr>
<td>dpb3x_driver_addr</td>
<td>+12</td>
<td>4</td>
<td>D</td>
<td>Pointer to driver</td>
</tr>
<tr>
<td>dpb3x_media</td>
<td>+16</td>
<td>1</td>
<td>B</td>
<td>Media byte</td>
</tr>
<tr>
<td>dpb3x_first_access</td>
<td>+17</td>
<td>1</td>
<td>B</td>
<td>This is initialized to -1 to force a media check the first time this DPB is used</td>
</tr>
<tr>
<td>dpb3x_next_dpb</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Pointer to next Drive parameter block</td>
</tr>
<tr>
<td>dpb3x_next_free</td>
<td>+1c</td>
<td>2</td>
<td>W</td>
<td>Cluster # of last allocated cluster</td>
</tr>
<tr>
<td>dpb3x_free_cnt</td>
<td>+1e</td>
<td>2</td>
<td>W</td>
<td>Count of free clusters, -1 if unknown</td>
</tr>
</tbody>
</table>

**DPB4X**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>

**Flag Definitions:**

- **DPBCHECK**: Disk in drive is being removed/checked for VPB.
- **DPBNONREMOV**: Drive supports non-removable media.
- **DPBVCRAMDISK**: Ram Disk Driver.
Current Directory Structure for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

SAS field SAS_file_CDS contains the selector for CDS RMP segment.

CDSAddr locates the RMP handle which contains the selector for the CDS RMP segment.

Locations

CDS segment is dynamically allocated from the kernel resident heap.

VM Owner
cdsrmp (0xff61).

Format
cddFATFS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cddFAT_id</td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>cluster of current dir</td>
</tr>
</tbody>
</table>
### cdfs

**Field Name** | Offset | Length | Type | Description
---|---|---|---|---
cdd_work[8] | 0 | 8 | S |

### cdksi

**Field Name** | Offset | Length | Type | Description
---|---|---|---|---
cdi_hVPB | 0 | 2 | W | hVPB for the drive mapped to this CDS

cdi_end | 2 | 2 | W | End of assignment

cdi_flags | 4 | 1 | B | fs independent flags (see below)
cdi_text[260] | 5 | 104 | A |

### curdir

**Field Name** | Offset | Length | Type | Description
---|---|---|---|---
cd_handle | 0 | 2 | W | lookup key for this CDS

cd_pid | 2 | 2 | W | PID part of lockup key for handles 1-26

cd_refcnt | 4 | 2 | W | reference count CDS's

cd_flags | 6 | 1 | B | See below for definitions

cd_devptr | 7 | 4 | D | local pointer to DPB or net device

cd_OwnerFSC | 15 | 2 | W | Owner FSC.Offst

cd_fsd | D | 8 | S | File system dependent section
cd_fsi | 15 | 10A | S | File system independent section
cdi_hVPB | 15 | 2 | W | hVPB for the drive mapped to this CDS
cdi_end | 17 | 2 | W | End of assignment
cdi_flags | 19 | 1 | B | fs independent flags (see below)
cdi_text[260] | 1A | 104 | B |

### cd_flags flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD_ISNET</td>
<td>0x80</td>
<td>This CDS is for a remote drive</td>
</tr>
<tr>
<td>CD_INUSE</td>
<td>0x40</td>
<td>This CDS is in use</td>
</tr>
<tr>
<td>CD_SPLICE</td>
<td>0x20</td>
<td>This CDS is for a JOINed drive</td>
</tr>
<tr>
<td>CD_JOIN</td>
<td>CD_SPLICE</td>
<td>This CDS is for a JOINed drive</td>
</tr>
<tr>
<td>CD_LOCAL</td>
<td>0x10</td>
<td>This CDS is for a SUBSTed drive</td>
</tr>
<tr>
<td>CD_ISPSEUDOCHAR</td>
<td>0x08</td>
<td>This CDS for a pseudo-char dev</td>
</tr>
<tr>
<td>CD_ISUNC</td>
<td>0x04</td>
<td>This CDS for a UNC name</td>
</tr>
</tbody>
</table>
cdi_flags flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDI_ISVALID</td>
<td>0x80</td>
<td>This CDS contains a valid cd_fsd</td>
</tr>
<tr>
<td>CDI_ISROOT</td>
<td>0x40</td>
<td>This CDS is for a root (no cdfsd)</td>
</tr>
<tr>
<td>CDI_MEDIASWAPPED</td>
<td>0x20</td>
<td>This CDS may not be valid (forces</td>
</tr>
</tbody>
</table>

File System Buffer for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

SAS field SAS_file_Buffers contains the selector for the file system buffer segment.

GDT_Buffers locates the GDT descriptor for the BUFSEG segment.

Locations

BUFSEG segment is dynamically allocated from the kernel resident heap.

VM Owner

fsbuf (0xff93).

Format

BUFSEG

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs_MRUHead</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>Head of MRU buffer list (LRU tail)</td>
</tr>
<tr>
<td>bs_MRUTail</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>Tail of MRU buffer list (LRU head)</td>
</tr>
<tr>
<td>bs_FreeHead</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Head of Free buffer list</td>
</tr>
<tr>
<td>bs_Handle</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Handle for virtual memory manager</td>
</tr>
<tr>
<td>bs_nBuffers</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>Number of buffers in segment</td>
</tr>
<tr>
<td>bs_buffsize</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>Size of buffer+header, in bytes.</td>
</tr>
<tr>
<td>bs_seglimit</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>Limit for entire buffer segment</td>
</tr>
<tr>
<td>bs_pStats</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>Offset of statistics block (for PROFILE)</td>
</tr>
<tr>
<td>bs_offRemMed</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>Minimum &quot;legal&quot; offset of buffer for removable media</td>
</tr>
<tr>
<td>bs_MaxSec</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>Maximum sector size for block device drivers</td>
</tr>
<tr>
<td>bs_BigBufBase</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>Base of big buffers pool</td>
</tr>
<tr>
<td>bs_BigBufMap</td>
<td>+16</td>
<td>2</td>
<td>W</td>
<td>Big buffers usage bit map (bit0 - Buf0)</td>
</tr>
<tr>
<td>bs_physBufSeg</td>
<td>+18</td>
<td>4</td>
<td>D</td>
<td>Buffer segment Physical Address</td>
</tr>
</tbody>
</table>

BUFFINFO
### Field Name | Offset | Length | Type | Description
---|---|---|---|---
buf_next | +0 | 2 | W | Pointer to next buffer in list (-1 = end)
buf_prev | +2 | 2 | W | Pointer to previous buffer in list (-1 = end)
buf_freeLink | +4 | 2 | W | Pointer to next free buffer (-1 = end)
buf_hVPB | +6 | 1 | W | Serial number of volume
buf_sector | +8 | 4 | D | Sector number of buffer
buf_wrtcnt | +c | 1 | B | For FAT sectors, # times sector written out
buf_wrtcntinc | +d | 2 | W | For FAT sectors, # sectors between each write
buf_flags | +f | 1 | B | Flags
buf_tid | +10 | 2 | W | Thread ID of buffer owner
buf_refcnt | +12 | 2 | W | Number of threads using buffer for read
buf_fill | +14 | 2 | W | Random debugging information
buf_pad | +16 | 2 | W | Force dword alignment.

### Buf_flags flag definitions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUF_DIRTY</td>
<td>0x80</td>
<td>Bit 7 = 1 if buffer dirty</td>
</tr>
<tr>
<td>BUF_VISIT</td>
<td>0x40</td>
<td>Bit 6 = 1 if buffer seen in search</td>
</tr>
<tr>
<td>BUF_WANT</td>
<td>0x20</td>
<td>Bit 5 = 1 if process waiting for buffer</td>
</tr>
<tr>
<td>BUF_BUSY</td>
<td>0x10</td>
<td>Bit 4 = 1 if in use by process</td>
</tr>
<tr>
<td>BUF_ISDATA</td>
<td>0x08</td>
<td>Bit 3 = 1 if buffer is DATA</td>
</tr>
<tr>
<td>BUF_ISDIR</td>
<td>0x04</td>
<td>Bit 2 = 1 if buffer is DIR</td>
</tr>
<tr>
<td>BUF_ISFAT</td>
<td>0x02</td>
<td>Bit 1 = 1 if buffer is FAT</td>
</tr>
<tr>
<td>BUF_ATTEMPTING_READ</td>
<td>0x01</td>
<td>Bit 0 = 1 if buffer is in swapper pool</td>
</tr>
</tbody>
</table>

-----------------------------

### Named Pipe Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

- **sf_sfd** points to the associated NP structure.
- **NmpRmpHand** locates the RMP handle that contains the selector for the NPN RMP segment.
- **npn_link** points to the double linked list of NP structures that are instances of the named pipe.

Instances of named pipes are double-lined by **np_flink** and **np_blink**.
NP fields `np_selector1` and `np_selector2` point to associated NPB structures.

Locations

The NPN RMP is allocated from the kernel swappable heap.

The NP is allocated from the system arena.

The NPB is allocated from the kernel resident heap.

VM Owner

NP owner id is `npipenp (0xff31)`.

NPN owner id is `npipenpn (0xff30)`.

NPB owner id is `npipenbuf (0xff9f)`.

Format

There are four important data structures associated with named pipes: the SFT corresponding to an open named pipe, a pair of kernel internal data structures describing the pipe and one or two allocated memory segments which contain the data buffers for the pipe.

The parts of the SFT specific to named pipes are:

```
sf_flags        SF_NMPIPE and SF_PIPE set
sf_np           pointer to pipe info.
sf_pipmod       mode of pipe, per-sft internal state bits
```

Where:

- `sf_np` is defined to be `sf_fsd+0`, the pointer to np structure
- `sf_pipmod` is defined to be `sf_fsd+4`, the mode of pipe, plus internal state

NP Named Pipe data structure

The internal data structure for an instance of a pipe. One of these structures is allocated for each open instance of a particular named pipe.

Allocated NP structures are placed on two lists. The first is headed by `ActiveNPLList`, with list pointer `np_next` linking together all currently active NP structures.

The second list is headed by the NPN structure defined below and is doubly-linked by the `np_flink` and `np_blink` pointers. This list is used to iterate over all instances of a particular pipe name.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>np_state</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>state of pipe</td>
</tr>
<tr>
<td>np_refcnt</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>SFT reference count for pipe (1 or 2)</td>
</tr>
<tr>
<td>np_next</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>pointer to next in active list</td>
</tr>
<tr>
<td>np_flink</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>pointer to next instance of pipe</td>
</tr>
<tr>
<td>np_blink</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>pointer to previous instance of pipe</td>
</tr>
<tr>
<td>np_namkey</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>RMP key value for npn structure</td>
</tr>
<tr>
<td>np_scnt</td>
<td>+a</td>
<td>1</td>
<td>B</td>
<td>count of servers (max. 1)</td>
</tr>
<tr>
<td>np_ccnt</td>
<td>+b</td>
<td>1</td>
<td>B</td>
<td>count of clients (max. 1)</td>
</tr>
<tr>
<td>np_selector1</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>selector for outgoing data buffer</td>
</tr>
<tr>
<td>np_selector2</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>selector for incoming data buffer</td>
</tr>
<tr>
<td>np_pipmod</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>pipe mode specified at creation time</td>
</tr>
<tr>
<td>np_flags</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>pipe flags</td>
</tr>
<tr>
<td>np_ssft</td>
<td>+14</td>
<td>4</td>
<td>D</td>
<td>back pointer to server SFT</td>
</tr>
</tbody>
</table>
NPN Named Pipe Name data structure

The following structure contains the common name for the multiple instances of a pipe. Its key value is used as the ProcBlock key for waiters on the pipe. The key value is also used as an RMP key to look up the name record from the NP structure.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npn_link</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>pointer to first instance</td>
</tr>
<tr>
<td>npn_key</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>unique serial number of name</td>
</tr>
<tr>
<td>npn_len</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>total length of structure</td>
</tr>
<tr>
<td>npn_name</td>
<td>+8</td>
<td>254</td>
<td>A</td>
<td>name of pipe, null terminated</td>
</tr>
</tbody>
</table>

NPB Named Pipe Buffer data structure

The following variables are used to control the access to a pipe buffer and are part of the allocated buffer for the pipe. In the case of a duplex pipe, two independent data buffers are allocated. Only one buffer will be allocated for a simplex pipe.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npb_selector</td>
<td>+0</td>
<td>2</td>
<td>W</td>
<td>selector of buffer</td>
</tr>
<tr>
<td>npb_first</td>
<td>+2</td>
<td>2</td>
<td>W</td>
<td>base of buffer</td>
</tr>
<tr>
<td>npb_in</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>next free byte in buffer</td>
</tr>
<tr>
<td>npb_out</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>next byte of data in buffer</td>
</tr>
<tr>
<td>npb_last</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>end+1 of buffer</td>
</tr>
<tr>
<td>npb_rdlck</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>read lock sem.</td>
</tr>
<tr>
<td>npb_wtlck</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>write lock sem.</td>
</tr>
<tr>
<td>npb_rdsem</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>read sync sem.</td>
</tr>
<tr>
<td>npb_wtsem</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>write sync sem.</td>
</tr>
<tr>
<td>npb_rdcnt</td>
<td>+12</td>
<td>1</td>
<td>B</td>
<td>count of readers of buffer</td>
</tr>
<tr>
<td>npb_wtcnt</td>
<td>+13</td>
<td>1</td>
<td>B</td>
<td>count of writers to buffer</td>
</tr>
<tr>
<td>npb_data</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>size of data left in pipe</td>
</tr>
</tbody>
</table>

np_state allowable values for named pipe state

Internally, byte stream mode pipes store just a collection of bytes in the data buffer. Message stream mode pipes have individual messages preceded by a word which indicates the size of the message.

Named pipes may be in one of several states depending on the actions that have been taken on it by the server end and client end. The following state/action table summarizes the valid state transitions:

<table>
<thead>
<tr>
<th>Current state</th>
<th>Action</th>
<th>Next state</th>
</tr>
</thead>
</table>
A special internal state, LISTEN2 is used when a client open is in progress (since some operations may block). This is treated the same as the LISTENING state except that a new open or wait will not recognize it as an available pipe.

If a server disconnects his end of the pipe, the client end will enter a special state in which any future operations (except close) on the file descriptor associated with the pipe will return an error.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP_DISCONNECTED</td>
<td>1</td>
<td>after pipe creation or Disconnect</td>
</tr>
<tr>
<td>NP_LISTENING</td>
<td>2</td>
<td>after DosNmPipeConnect</td>
</tr>
<tr>
<td>NP_CONNECTED</td>
<td>3</td>
<td>after Client open</td>
</tr>
<tr>
<td>NP_CLOSING</td>
<td>4</td>
<td>after Client close</td>
</tr>
<tr>
<td>NP_LISTEN2</td>
<td>0x12</td>
<td>internal; client open in progress</td>
</tr>
</tbody>
</table>

**np_pipmod, sf_pipmod** bit mask values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP_NBLK</td>
<td>0x8000</td>
<td>non-blocking read/write</td>
</tr>
<tr>
<td>NP_NBLKR</td>
<td>0x8000</td>
<td>non-blocking read</td>
</tr>
<tr>
<td>NP_NBLKW</td>
<td>0x8000</td>
<td>non-blocking write</td>
</tr>
<tr>
<td>NP_SERVER</td>
<td>0x4000</td>
<td>set if server end</td>
</tr>
<tr>
<td>NP_WMESG</td>
<td>0x0400</td>
<td>write messages</td>
</tr>
<tr>
<td>NP_RMMSG</td>
<td>0x0100</td>
<td>read as messages</td>
</tr>
<tr>
<td>NP_TIMOUT</td>
<td>0x3800</td>
<td>Timeout np_sem_blk &amp; np_sem_wait</td>
</tr>
</tbody>
</table>

--------------------------------------------

Anonymous Pipe Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0

Points

SFT field sfi_hVPB contains the selector that maps IOBLOCK structure.

Locations

The pipe IOBLOCK is allocated from the kernel heaps.

VM Owner

pipe (0xfff80).

Format
**Anonymous Pipe data structure**

A 'pipe' is a connection between (among) file handles (JFN's). Data written to the 'write end' of the pipe are made available for reading on the 'read end'. The `pipe` system call creates a pipe and returns two file handles, one for the read end and one for the write end. These handles are manipulated in the same way as normal file handles; they may be 'dup'ed and are inherited in the same way. Data are written into a pipe via a 'write' system call on the write end of the pipe. Likewise, data are read from the pipe via a 'read' call on the read end.

Data that are written to a pipe are captured in a circular buffer. The size of the buffer is specified when the pipe is created; if no size is specified, a default size is used.

The circular buffer is described by an 'ioblock'. The ioblock is the buffer's header; the circular buffer proper follows the ioblock in a heap memory object (mapped by a GDT selector) allocated when the pipe is created. The ioblock contains all of the per-pipe information, such as reader, writer, and reference counts, and also holds the pointers into the circular buffer proper.

The selector that points to the circular buffer is stored in the SFT, at sft_hVPB.

When the in and out pointers are equal, the circular buffer is empty. When the in pointer trails the out pointer by 1, the buffer is full. Thus, a 512 byte buffer can hold only 511 bytes; one byte is lost so that full and empty conditions can be distinguished. So that the user can put 512 bytes in a pipe that they created with a size of 512, we allow for this byte lost when allocating the segment.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>io_inprogcnt</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>count of read/writes in progress</td>
</tr>
<tr>
<td>io_refcnt</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>count of references</td>
</tr>
<tr>
<td>io_rdrcnt</td>
<td>+2</td>
<td>1</td>
<td>B</td>
<td>count of readers</td>
</tr>
<tr>
<td>io_wtrcnt</td>
<td>+3</td>
<td>1</td>
<td>B</td>
<td>count of writers</td>
</tr>
<tr>
<td>io_selector</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>buffer selector</td>
</tr>
<tr>
<td>io_first</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>ptr to base of circular buffer</td>
</tr>
<tr>
<td>io_in</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>ptr to next free byte</td>
</tr>
<tr>
<td>io_out</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>ptr to next byte of data</td>
</tr>
<tr>
<td>io_last</td>
<td>+c</td>
<td>2</td>
<td>W</td>
<td>ptr to end+1 of buffer</td>
</tr>
<tr>
<td>io_rdiksem</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>read lock semaphore</td>
</tr>
<tr>
<td>io_wtksem</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>write lock semaphore</td>
</tr>
<tr>
<td>io_rdssem</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>read sync semaphore</td>
</tr>
<tr>
<td>io_wtssem</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>write sync semaphore</td>
</tr>
</tbody>
</table>

--------------------------------------------

**I/O System Control Block Reference**

The following control blocks are described in this section:

- **Physical Device Driver Header (DEV)**
- **PDD IRQ Information Blocks (DIRQ)**
- **Virtual Device Driver Entry Point Structures**
- **Device Driver Request Packets (REQ)**
- **BIOS Parameter Block (BPB)**

An overview of the I/O System Control Blocks follows:

---------------------------------------------
I/O System Control Block Diagrams

The following diagrams illustrate the relationships between various I/O system control blocks:

- Physical Device Driver Communication
- Physical Device Driver IRQ Sharing
- Virtual Device Driver Communication

---------------------------------------------

Physical Device Driver Communication
Physical Device Driver Communication

SAS

SAS device driver section

PDD1

Strategy ep
IDC ep
Name

FDD/VED ep

PDD2

Name

Name

PDD3

Name

_pddephead

PDDEP

Registered Name

PDDEP

Registered Name
Physical Device Driver IRQ Sharing
Physical Device Driver IRQ Sharing

IRQ array

\[ \text{DIRQ} \rightarrow \text{PDD1} \]
\[ \text{Interrupt entry pt} \]

\[ \text{DIRQ} \rightarrow \text{PDD2} \]
\[ \text{Interrupt entry pt} \]
Virtual Device Driver Communication
Virtual Device Driver Communication

VDDEP

Registered Name

MTE

VDDEP

VDs registering state name

pyddepm

VDDPROC

VDD/VDD ev

VDD/App ev

HDLVDD

_phyVDDread_
Physical Device Driver Header (DEV) for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

DPB field `dpb_driver_addr` points to the associated Physical Device Driver Header.

SFT field `sf_devptr` points to the associated Physical Device Driver Header.

Locations

Built at the beginning of the first module segment of the device driver.

VM Owner

`dd1 (0xff50) to dd16 (0xff5f)`.

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDevNext</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Pointer to next device header</td>
</tr>
<tr>
<td>SDevAtt</td>
<td>+4</td>
<td>2</td>
<td>W</td>
<td>Attributes of the device</td>
</tr>
<tr>
<td>SDevStrat</td>
<td>+6</td>
<td>2</td>
<td>W</td>
<td>Strategy entry point</td>
</tr>
<tr>
<td>SDevInt</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>IDC entry point</td>
</tr>
<tr>
<td>SDevName</td>
<td>+a</td>
<td>8</td>
<td>A</td>
<td>name (block uses only 1st byte)</td>
</tr>
<tr>
<td>SDevProtCS</td>
<td>+12</td>
<td>2</td>
<td>W</td>
<td>Protect-mode CS of strategy entry pt</td>
</tr>
<tr>
<td>SDevProtDS</td>
<td>+14</td>
<td>2</td>
<td>W</td>
<td>Protect-mode DS</td>
</tr>
<tr>
<td>SDevRealCS</td>
<td>+16</td>
<td>2</td>
<td>W</td>
<td>Real-mode CS of strategy entry pt</td>
</tr>
<tr>
<td>SDevRealDS</td>
<td>+18</td>
<td>2</td>
<td>W</td>
<td>Real-mode DS</td>
</tr>
<tr>
<td>SDevCaps</td>
<td>+20</td>
<td>4</td>
<td>D</td>
<td>bit map of DD/MM restrictions</td>
</tr>
</tbody>
</table>

`SDevCaps` flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV_IOCTL2</td>
<td>0x0001</td>
<td>DD can handle dev ioclt2</td>
</tr>
<tr>
<td>DEV_16MB</td>
<td>0x0002</td>
<td>DD can handle phys.addresses &gt;16MB</td>
</tr>
<tr>
<td>DEV_PARALLEL</td>
<td>0x0004</td>
<td>DD handles parallel port</td>
</tr>
<tr>
<td>DEV_ADAPTER_DD</td>
<td>0x0008</td>
<td>DD supports Adapter Dev Driver Intf</td>
</tr>
<tr>
<td>DEV_INITCOMPLETE</td>
<td>0x0010</td>
<td>DD can handle CMDInitComplete</td>
</tr>
</tbody>
</table>

Device Driver Type definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV_CIN</td>
<td>0x0001 025</td>
<td>Device is console in</td>
</tr>
</tbody>
</table>
Level definitions for devices

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVLEV_0</td>
<td>0x0000</td>
<td>DOS 3.0 and before</td>
</tr>
<tr>
<td>DEVLEV_1</td>
<td>0x0080</td>
<td>DOS 5.0</td>
</tr>
<tr>
<td>DEVLEV_2</td>
<td>0x0100</td>
<td>OS/2 v1.2 (new gen ioctl iface)</td>
</tr>
<tr>
<td>DEVLEV_3</td>
<td>0x0180</td>
<td>OS/2 v2.0 (support of memory above 16MB)</td>
</tr>
</tbody>
</table>

PDD IRQ Information Blocks (DIRQ) for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

IRQI field irqi_pdirqHead points to the head of a chain of associated DIRQs.

Locations

airqi locates the table of IRQI entries.

DIRQs are allocated dynamically from the kernel resident heap.

The IRQI array is a static part of the OS2KRNL load module.

VM Owners

IRQI owner id: os2krnl (0xffaa).

DIRQ owner id: intdirq (0xff78).

Format

IRQI

Field Name Offset Length Type Description
irqi_pdirqHead +0 4 D Head of shared DD chain (0 = not set)
irqi_usIRQNum +4 2 W IRQ number
irqi_usFlags +6 2 W IRQ Flags

irqi_usFlags flag definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>irqf_fVDM</td>
<td>0x0004</td>
<td>If set, this IRQ is a candidate for routing to a VDM, if it is not claimed by a PDD</td>
</tr>
<tr>
<td>irqf_fNPX</td>
<td>0x0008</td>
<td>If set, the IRQ is the NPX interrupt level</td>
</tr>
<tr>
<td>irqf_fSharing</td>
<td>0x0010</td>
<td>If set, the IRQ is sharable. If clear the IRQ can not be shared by DD.</td>
</tr>
<tr>
<td>irqf_fSys</td>
<td>0x0020</td>
<td>If set, the IRQ is owned by the system and the handler can not be changed or removed by a device driver. Set initially for the slave, IRQ 2.</td>
</tr>
<tr>
<td>irqf_fShared</td>
<td>0x0040</td>
<td>If set, the IRQ can be shared by more than 1 DD. This bit reflects the shared parameter of the first dh_SetIRQ issued for this level.</td>
</tr>
</tbody>
</table>

DIRQ

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirq_pdirqLink</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Next DIRQ structure in list</td>
</tr>
<tr>
<td>dirq_f16pfn</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>DD’s interrupt handler</td>
</tr>
<tr>
<td>dirq_usDS</td>
<td>+8</td>
<td>2</td>
<td>W</td>
<td>DD’s data segment</td>
</tr>
<tr>
<td>dirq_usIRQNum</td>
<td>+a</td>
<td>2</td>
<td>W</td>
<td>IRQ number</td>
</tr>
<tr>
<td>dirq_pdirqFreeList</td>
<td>+c</td>
<td>4</td>
<td>D</td>
<td>list of unset DIRQs</td>
</tr>
</tbody>
</table>

---------------------------------------------------------------

Virtual Device Driver Entry Point Structures

Pointers

_pvddepHead points to the head of a chain of VDDEP structures. One is allocated for each VDD the registers an either or both a VDD/VDD or VDD/OS2 entry point. There entry points are used respectively when either a VDHRequestVDD/VDHOpenVDD or DosRequestVDD/DosOpenVDD call is made.

VDDEP field vddep_vddp points to the associated chain of VDDPROC structures. One is allocated for each VDD the registers entry points under the same name.

_phdlVddHead points to the head of a chain of HDLVDD structures. One is allocated for each open VDD. The handle returned is the address of the associated HDLVDD.
_ppddephead points to the head of a chain of PDDEP structures. One is allocated for each Physical Device Driver that registers an entry point for VDD/PDD communication. The entry point is registered using DevHlp_RegisterPDD, and accessed using VDHRequestPDD.

Locations
VDDEPs, VDDPROCs, HDLVDDs and PDDEPs are allocated dynamically from the kernel resident heap.

VM Owner
VDDEP owner id: vddep (0xffd2).
VDDPROC owner id: vddproc (0xffdb).
HDLVDD owner id: vddlvr (0xffd7).
PDDEP owner id: vddpddep (0xffda).

Format

**VDDEP**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vddep_szVDD</td>
<td>+0</td>
<td>9</td>
<td>A</td>
<td>VDD Name</td>
</tr>
<tr>
<td>vddep_vddp</td>
<td>+9</td>
<td>4</td>
<td>D</td>
<td>VDD entry points (pointer to VDDPROC)</td>
</tr>
<tr>
<td>vddep_hmte</td>
<td>+d</td>
<td>4</td>
<td>D</td>
<td>VDD hmte for deregistering if VDD fails</td>
</tr>
<tr>
<td>vddep_pvddep</td>
<td>+11</td>
<td>4</td>
<td>D</td>
<td>Next VDD (pointer to next VDDEP)</td>
</tr>
</tbody>
</table>

**VDDPROC**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vddproc_pfnvdd</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>Entry point for VDD/VDD comm.</td>
</tr>
<tr>
<td>vddproc_pfnos2</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Entry point for OS2/VDD comm.</td>
</tr>
<tr>
<td>pvddproc</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Entry points registered with same name</td>
</tr>
</tbody>
</table>

**HDLVDD**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdlvdd_pvddproc</td>
<td>+0</td>
<td>4</td>
<td>D</td>
<td>VDD routine to be called (pointer to VDDPROC)</td>
</tr>
<tr>
<td>hdlvdd</td>
<td>+4</td>
<td>4</td>
<td>D</td>
<td>Pointer to next VDD handle; NULL if no more</td>
</tr>
</tbody>
</table>

**PDDEP**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pddep_szPDD</td>
<td>+0</td>
<td>9</td>
<td>A</td>
<td>PDD name</td>
</tr>
<tr>
<td>pddep_fpfn</td>
<td>+9</td>
<td>4</td>
<td>D</td>
<td>Entry point routine</td>
</tr>
<tr>
<td>pddep_ppddep</td>
<td>+d</td>
<td>4</td>
<td>D</td>
<td>Next entry point (PDDEP)</td>
</tr>
</tbody>
</table>

---------------------------------------------------------------
Device Driver (Strategy 1) Request Packet (REQ) for OS/2 Warp V4.0 and OS/2 Warp V3.0

Pointers

TCB field **TCBReqPkt** points to the Request Packet pre-allocated to a thread.

Locations

Allocated from the Request Packet Pool in the System Arena.

VM Owner

requktl (0xff40).

Format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>+0</td>
<td>20</td>
<td>S</td>
<td>Device Driver Request Packet</td>
</tr>
<tr>
<td>PktLen</td>
<td>+0</td>
<td>1</td>
<td>B</td>
<td>length in bytes of packet</td>
</tr>
<tr>
<td>PktUnit</td>
<td>+1</td>
<td>1</td>
<td>B</td>
<td>subunit number of block device</td>
</tr>
<tr>
<td>PktCmd</td>
<td>+2</td>
<td>1</td>
<td>B</td>
<td>command code</td>
</tr>
<tr>
<td>PktStatus</td>
<td>+3</td>
<td>2</td>
<td>W</td>
<td>status word</td>
</tr>
<tr>
<td>PktFlag</td>
<td>+5</td>
<td>1</td>
<td>B</td>
<td>disk driver internal flags</td>
</tr>
<tr>
<td>PktFlag</td>
<td>+6</td>
<td>3</td>
<td>B</td>
<td>reserved</td>
</tr>
<tr>
<td>PktDOSLink</td>
<td>+5</td>
<td>4</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>PktDevLink</td>
<td>+9</td>
<td>4</td>
<td>D</td>
<td>device multiple-request link</td>
</tr>
<tr>
<td>PktData</td>
<td>+d</td>
<td>18</td>
<td>S</td>
<td>data pertaining to specific packet</td>
</tr>
<tr>
<td>PktData</td>
<td>d</td>
<td>10</td>
<td>S</td>
<td>Generic IOCTL</td>
</tr>
<tr>
<td>GIOCATEGORY</td>
<td>+d</td>
<td>1</td>
<td>B</td>
<td>Category Code</td>
</tr>
<tr>
<td>GIOFunction</td>
<td>+e</td>
<td>1</td>
<td>B</td>
<td>Function code</td>
</tr>
<tr>
<td>GIOparaPack</td>
<td>+f</td>
<td>4</td>
<td>D</td>
<td>pointer to parameter packet</td>
</tr>
<tr>
<td>GIODataPack</td>
<td>+13</td>
<td>4</td>
<td>D</td>
<td>pointer to data packet</td>
</tr>
<tr>
<td>GIOSFN</td>
<td>+17</td>
<td>2</td>
<td>W</td>
<td>(used by Spooler?)</td>
</tr>
<tr>
<td>GIOParaLen</td>
<td>+19</td>
<td>2</td>
<td>W</td>
<td>length of parameter packet</td>
</tr>
<tr>
<td>GIODataLen</td>
<td>+1b</td>
<td>2</td>
<td>W</td>
<td>length of data packet</td>
</tr>
<tr>
<td>GIOParaLen</td>
<td>d</td>
<td>c</td>
<td>S</td>
<td>INIT Command for Base DDs (0 and 27)</td>
</tr>
<tr>
<td>InitcUnit</td>
<td>+d</td>
<td>1</td>
<td>B</td>
<td>number of units returned</td>
</tr>
<tr>
<td>InitpEnd</td>
<td>+e</td>
<td>4</td>
<td>D</td>
<td>pointer to free mem after dev</td>
</tr>
<tr>
<td>InitDevHlp</td>
<td>+e</td>
<td>4</td>
<td>D</td>
<td>address of Device Helper router</td>
</tr>
<tr>
<td>InitEcode</td>
<td>+e</td>
<td>2</td>
<td>W</td>
<td>size of code segment</td>
</tr>
<tr>
<td>InitEdata</td>
<td>+10</td>
<td>2</td>
<td>W</td>
<td>size of data segment</td>
</tr>
<tr>
<td>InitParms</td>
<td>+12</td>
<td>4</td>
<td>D</td>
<td>pointer parameters</td>
</tr>
</tbody>
</table>
InitpBPB +12 4 D  pointer to BPBs
Initdrv +16 1 B  drive no. assigned to unit 0
          +17 1 B  reserved
InitSysiData +18 1 B  SysInit's DOSALIAS selector
          (for resident drivers only)
          +d  d  S  query for extended capability command (0x1d)
          +d  3  B  reserved
DCS_Addr +10 4 W  16
VCS_Addr +14 4 W  16
          +d  6  B  Media Check command 1
MedChkmedia +d  1  B  last media byte seen
MedChkflaga +e  1  B  -1=change 0=dont know 1=no change
MedChkpVIDa +f  4  D  pointer to VID
          +d  9  S  build BPB command 2
BldBPBmedia +d  1  B  media byte
BldBPBbuffer +e  4  D  scratch buffer
          +d  f  S  Read/Write IO commands 3, 4, 8, 9, 12, 24, 25, 26
IOmedia +d  1  B  media byte
IOpData +e  4  D  transfer address
IOcount +12 2 W  count of bytes/sectors
IOstart +14 2 W  starting sector (block)
IOPhysRBA +14 4 D  physical starting sector
IOSFNsRBA +18 2 W  for device only
PktAdvise +la  2 W  for >= v12 only
          +d  4  S  Device Open/Close commands 13 and 14
OCSFN +d  2 W  sfn of open instance for virtualization
          +d  1  S  Start/Stop console commands (98, 99)
CStpSKG +d  1  B  Screen/Keyboard number
          +d  6  S  De-install driver command 20
DINEndLocn +d  4  D
DINLengthn +11 2 W

PktStatus word masks
### Name | Bit Mask | Description
---|---|---
STERR | 0x8000 | Bit 15 - Error
STINTER | 0x0400 | Bit 10 - Interim character
STBUI | 0x0200 | Bit 9 - Busy
STDON | 0x0100 | Bit 8 - Done
STECODE | 0x00ff | Error code
WRECODE | 0

#### PktFlag flags

<table>
<thead>
<tr>
<th>Name</th>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fPktInt13RP</td>
<td>0x01</td>
<td>Int 13 Request Packet</td>
</tr>
<tr>
<td>fPktCallOutDone</td>
<td>0x02</td>
<td>Int 13 Callout completed</td>
</tr>
<tr>
<td>fPktDiskIOTchd</td>
<td>0x04</td>
<td>Disk_IO has touched this packet</td>
</tr>
<tr>
<td>STDON</td>
<td>0x0100</td>
<td>Bit 8 - Done</td>
</tr>
<tr>
<td>STECODE</td>
<td>0x00ff</td>
<td>Error code</td>
</tr>
</tbody>
</table>
| WRECODE                  | 0

See [Device Driver Strategy Commands](#) for a cross-reference of PktCmd command codes.

---------------------------

**BIOS Parameter Block (BPB) for OS/2 Warp V4.0 and OS/2 Warp V3.0**

#### Pointers

I/O Request Packet fields **InitpBPB** and **BldBPBuffer** point to the BPB structure.

#### Locations

Allocated from the System Arena.

#### VM Owner

Non-specific.

#### Format

**BPB**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPSECSZ</td>
<td>0</td>
<td>2</td>
<td>W</td>
<td>Size in bytes of physical sector</td>
</tr>
<tr>
<td>BPCLUS</td>
<td>2</td>
<td>1</td>
<td>B</td>
<td>Sectors/Alloc unit</td>
</tr>
<tr>
<td>BPRES</td>
<td>3</td>
<td>2</td>
<td>W</td>
<td>Number of reserved sectors</td>
</tr>
<tr>
<td>BPFTCNT</td>
<td>5</td>
<td>1</td>
<td>B</td>
<td>Number of FATs</td>
</tr>
<tr>
<td>BPRDCNT</td>
<td>6</td>
<td>2</td>
<td>W</td>
<td>Number of directory entries</td>
</tr>
</tbody>
</table>
Reference Tables

The following reference information is tabulated in this section:

- OS/2 System Error Codes
- OS/2 System Exception Codes
- Trap Screen Reference
- Standard GDT Assignments
- Standard LDT Assignments
- VM System Owner Ids
- DevHlp Function Cross-reference
- Device Driver Strategy Commands
- System Ordinal Cross-reference
- OS/2 Fix Pack to Build Level Cross-reference

System Error Codes

OS/2 System Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NO ERROR</td>
</tr>
<tr>
<td>1</td>
<td>INVALID FUNCTION</td>
</tr>
<tr>
<td>2</td>
<td>FILE NOT FOUND</td>
</tr>
<tr>
<td>3</td>
<td>PATH NOT FOUND</td>
</tr>
</tbody>
</table>
4        TOO MANY OPEN FILES
5        ACCESS DENIED
6        INVALID HANDLE
7        ARENA TRASHED
8        NOT ENOUGH MEMORY
9        INVALID BLOCK
10       BAD ENVIRONMENT
11       BAD FORMAT
12       INVALID ACCESS
13       INVALID DATA
15       INVALID DRIVE
16       CURRENT DIRECTORY
17       NOT SAME DEVICE
18       NO MORE FILES
19       WRITE PROTECT
20       BAD UNIT
21       NOT READY
22       BAD COMMAND
23       CRC
24       BAD LENGTH
25       SEEK
26       NOT DOS DISK
27       SECTOR NOT FOUND
28       OUT OF PAPER
29       WRITE FAULT
30       READ FAULT
31       GEN FAILURE
32       SHARING VIOLATION
33       LOCK VIOLATION
34       WRONG DISK
35       FCB UNAVAILABLE
36       SHARING BUFFER EXCEEDED
37       CODE PAGE MISMATCHED
38       HANDLE EOF
39       HANDLE DISK FULL
40       BAD COMMAND
41       CRC
42       BAD LENGTH
43       SEEK
44       NOT DOS DISK
SECTOR NOT FOUND
OUT OF PAPER
WRITE FAULT
READ FAULT
GEN FAILURE
NOT SUPPORTED
REM NOT LIST
DUP NAME
BAD NETPATH
NETWORK BUSY
DEV NOT EXIST
TOO MANY CMDS
ADAP HDW ERR
BAD NET RESP
UNEXP NET ERR
BAD REM ADAP
PRINTQ FULL
NO SPOOL SPACE
PRINT CANCELLED
NETNAME DELETE
NETWORK ACCESS DENIED
BAD DEV TYPE
BAD NET NAME
TOO MANY NAMES
TOO MANY SESS
SHARING PAUSED
REQ NOT ACCEP
REDIR PAUSED
SBCS ATT WRITE PROT
SBCS GENERAL FAILURE
XGA OUT MEMORY
FILE EXISTS
DUP FCB
CANNOT MAKE
FAIL I24
OUT OF STRUCTURES
ALREADY ASSIGNED
INVALID PASSWORD
INVALID PARAMETER
88       NET WRITE FAULT
89       NO PROC SLOTS
90       NOT FROZEN
91       SYS COMP NOT LOADED
92       ERR TSTOVFL
93       ERR TSTDUP
94       NO ITEMS
95       INTERRUPT
96       INVALID DTA
97       DEVICE IN USE
98       TOO MANY SEMAPHORES
99       EXCL SEM ALREADY OWNED
100      SEM IS SET
101      TOO MANY SEM REQUESTS
102      INVALID AT INTERRUPT TIME
103      SEM OWNER DIED
104      SEM USER LIMIT
105      DISK CHANGE
106      DRIVE LOCKED
107      BROKEN PIPE
108      OPEN FAILED
109      BUFFER OVERFLOW
110      DISK FULL
111      NO MORE SEARCH HANDLES
112      INVALID TARGET HANDLE
113      PROTECTION VIOLATION
114      VIOKBD REQUEST
115      INVALID CATEGORY
116      INVALID VERIFY SWITCH
117      BAD DRIVER LEVEL
118      CALL NOT IMPLEMENTED
119      SEM TIMEOUT
120      INSUFFICIENT BUFFER
121      INVALID NAME
122      HPFS INVALID VOLUME CHAR
123      INVALID LEVEL
124      NO VOLUME LABEL
125      MOD NOT FOUND
126      PROC NOT FOUND
WAIT NO CHILDREN
CHILD NOT COMPLETE
DIRECT ACCESS HANDLE
NEGATIVE SEEK
SEEK ON DEVICE
IS JOIN TARGET
IS JOINED
IS SUBSTED
NOT JOINED
NOT SUBSTED
JOIN TO JOIN
SUBST TO SUBST
JOIN TO SUBST
SUBST TO JOIN
BUSY DRIVE
SAME DRIVE
DIR NOT ROOT
DIR NOT EMPTY
IS SUBST PATH
IS JOIN PATH
PATH BUSY
IS SUBST TARGET
SYSTEM TRACE
INVALID EVENT COUNT
TOO MANY MUXWAITERS
INVALID LIST FORMAT
LABEL TOO LONG
HPFS VOL LABEL LONG
TOO MANY TCBS
SIGNAL REFUSED
DISCARDED
NOT LOCKED
BAD THREADID ADDR
BAD ARGUMENTS
BAD PATHNAME
SIGNAL PENDING
UNCERTAIN MEDIA
MAX THRDS REACHED
MONITORS NOT SUPPORTED
UNC DRIVER NOT INSTALLED
LOCK FAILED
SWAPIO FAILED
SWAPIN FAILED
BUSY
INT TOO LONG
CANCEL VIOLATION
ATOMIC LOCK NOT SUPPORTED
READ LOCKS NOT SUPPORTED
INVALID SEGMENT NUMBER
INVALID CALLGATE
INVALID ORDINAL
ALREADY EXISTS
NO CHILD PROCESS
CHILD ALIVE NOWAIT
INVALID FLAG NUMBER
SEM NOT FOUND
INVALID STARTING CODESEG
INVALID STACKSEG
INVALID MODULETYPE
INVALID EXE SIGNATURE
EXE MARKED INVALID
BAD EXE FORMAT
ITERATED DATA EXCEEDS 64k
INVALID MINALLOCsize
DYNLINK FROM INVALID RING
IOPL NOT ENABLED
INVALID SEGDPL
AUTODATESEG EXCEEDS 64k
RING2SEG MUST BE MOVABLE
RELOC CHAIN XEEDS SEGLIM
INFLOOP IN RELOC CHAIN
ENVVAR NOT FOUND
NOT CURRENT CTRY
NO SIGNAL SENT
FILENAME EXCEDE RANGE
RING2 STACK IN USE
META EXPANSION TOO LONG
INVALID SIGNAL NUMBER
THREAD 1 INACTIVE
INFO NOT AVAIL
LOCKED
BAD DYNALINK
TOO MANY MODULES
NESTING NOT ALLOWED
CANNOT SHRINK
ZOMBIE PROCESS
STACK IN HIGH MEMORY
INVALID EXITROUTINE RING
GETBUF FAILED
FLUSHBUF FAILED
TRANSFER TOO LONG
FORCENOSWAP FAILED
SMG NO TARGET WINDOW
NO CHILDREN
INVALID SCREEN GROUP
BAD PIPE
PIPE BUSY
NO DATA
PIPE NOT CONNECTED
MORE DATA
VC DISCONNECTED
CIRCULARITY REQUESTED
DIRECTORY IN CDS
INVALID FSD NAME
INVALID PATH
INVALID EA NAME
EA LIST INCONSISTENT
EA LIST TOO LONG
NO META MATCH
FINDNOTIFY TIMEOUT
NO MORE ITEMS
SEARCH STRUC REUSED
CHAR NOT FOUND
TOO MUCH STACK
INVALID ATTR
INVALID STARTING RING
INVALID DLL INIT RING
CANNOT COPY
DIRECTORY
268 OPLOCKED FILE
269 OPLOCK THREAD EXISTS
270 VOLUME CHANGED
271 FINDNOTIFY HANDLE IN USE
272 FINDNOTIFY HANDLE CLOSED
273 NOTIFY OBJECT REMOVED
274 ALREADY SHUTDOWN
275 EAS DIDNT FIT
276 EA FILE CORRUPT
277 EA TABLE FULL
278 INVALID EA HANDLE
279 NO CLUSTER
280 CREATE EA FILE
281 CANNOT OPEN EA FILE
282 EAS NOT SUPPORTED
283 NEED EAS FOUND
284 DUPLICATE HANDLE
285 DUPLICATE NAME
286 EMPTY MUXWAIT
287 MUTEX OWNED
288 NOT OWNER
289 PARAM TOO SMALL
290 TOO MANY HANDLES
291 TOO MANY OPENS
292 WRONG TYPE
293 UNUSED CODE
294 THREAD NOT TERMINATED
295 INIT ROUTINE FAILED
296 MODULE IN USE
297 NOT ENOUGH WATCHPOINTS
298 TOO MANY POSTS
299 ALREADY POSTED
300 ALREADY RESET
301 SEM BUSY
302 INVALID PROCID
303 INVALID PDELTA
304 NOT DESCENDANT
305 NOT SESSION MANAGER
306 INVALID PCLASS
INVALID SCOPE
INVALID THREADID
DOSSUB SHRINK
DOSSUB NOMEM
DOSSUB OVERLAP
DOSSUB BADSIZE
DOSSUB BADFLAG
DOSSUB BADSELECTOR
MR MSG TOO LONG
MGS MR MSG TOO LONG
MR MID NOT FOUND
MR UN ACC MSGF
MR INV MSGF FORMAT
MR INV IVCOUNT
MR UN PERFORM
TS WAKEUP
TS SEMHANDLE
TS NOTIMER
TS HANDLE
TS DATETIME
SYS INTERNAL
QUE CURRENT NAME
QUE PROC NOT OWNED
QUE PROC OWNED
QUE DUPLICATE
QUE ELEMENT NOT EXIST
QUE NO MEMORY
QUE INVALID NAME
QUE INVALID PRIORITY
QUE INVALID HANDLE
QUE LINK NOT FOUND
QUE MEMORY ERROR
QUE PREV AT END
QUE PROC NO ACCESS
QUE EMPTY
QUE NAME NOT EXIST
QUE NOT INITIALIZED
QUE UNABLE TO ACCESS
QUE UNABLE TO ADD
QUE UNABLE TO INIT
VIO INVALID MASK
VIO PTR
VIO APTR
VIO RPTR
VIO CPTR
VIO LPTR
VIO MODE
VIO WIDTH
VIO ATTR
VIO ROW
VIO COL
VIO TOPROW
VIO BOTROW
VIO RIGHTCOL
VIO LEFTCOL
SCS CALL
SCS VALUE
VIO WAIT FLAG
VIO UNLOCK
SGS NOT SESSION MGR
SMG INVALID SGID
SMG INVALID SESSION ID
SMG NOSG
SMG NO SESSIONS
SMG GRP NOT FOUND
SMG SESSION NOT FOUND
SMG SET TITLE
KBD PARAMETER
KBD NO DEVICE
KBD INVALID IOWAIT
KBD INVALID LENGTH
KBD INVALID ECHO MASK
KBD INVALID INPUT MASK
KBD INVALID INPUT MASK
MON INVALID PARMS
MON INVALID DEVNAME
MON INVALID HANDLE
MON BUFFER TOO SMALL
MON BUFFER EMPTY
MON DATA TOO LARGE
MOUSE NO DEVICE
MOUSE INV HANDLE
MOUSE INV PARMS
MOUSE CANT RESET
MOUSE DISPLAY PARMS
MOUSE INV MODULE
MOUSE INV ENTRY PT
MOUSE INV MASK
NO MOUSE NO DATA
NO MOUSE PTR DRAWN
INVALID FREQUENCY
NLS NO COUNTRY FILE
NO COUNTRY SYS
NLS OPEN FAILED
OPEN COUNTRY SYS
NLS NO CTRY CODE
NO COUNTRY OR CODEPAGE
NLS TABLE TRUNCATED
NLS BAD TYPE
NLS TYPE NOT FOUND
COUNTRY NO TYPE
VIO SMG ONLY
VIO INVALID ASCIIZ
VIO DEREGISTER
VIO NO POPUP
VIO EXISTING POPUP
KBD SMG ONLY
KBD INVALID ASCIIZ
KBD INVALID MASK
KBD REGISTER
KBD DEREGISTER
MOUSE SMG ONLY
MOUSE INVALID ASCIIZ
MOUSE INVALID MASK
MOUSE REGISTER
MOUSE DEREGISTER
SMG BAD ACTION
SMG INVALID CALL
SCS SG NOTFOUND
420      SCS NOT SHELL
421      VIO INVALID PARMS
422      VIO FUNCTION OWNED
423      VIO RETURN
424      SCS INVALID FUNCTION
425      SCS NOT SESSION MGR
426      VIO REGISTER
427      VIO NO MODE THREAD
428      VIO NO SAVE RESTORE THD
429      VIO IN BG
430      VIO ILLEGAL DURING POPUP
431      SMG NOT BASESHELL
432      SMG BAD STATUSREQ
433      QUE INVALID WAIT
434      VIO LOCK
435      MOUSE INVALID IOWAIT
436      VIO INVALID HANDLE
437      VIO ILLEGAL DURING LOCK
438      VIO INVALID LENGTH
439      KBD INVALID HANDLE
440      KBD NO MORE HANDLE
441      KBD CANNOT CREATE KCB
442      KBD CODEPAGE LOAD INCOMPL
443      KBD INVALID CODEPAGE ID
444      KBD NO CODEPAGE SUPPORT
445      KBD FOCUS REQUIRED
446      KBD FOCUS ALREADY ACTIVE
447      KBD KEYBOARD BUSY
448      KBD INVALID CODEPAGE
449      KBD UNABLE TO FOCUS
450      SMG SESSION NON SELECT
451      SMG SESSION NOT FOREGRND
452      SMG SESSION NOT PARENT
453      SMG INVALID START MODE
454      SMG INVALID RELATED OPT
455      SMG INVALID BOND OPTION
456      SMG INVALID SELECT OPT
457      SMG START IN BACKGROUND
458      SMG INVALID STOP OPTION
SMG BAD RESERVE
SMG PROCESS NOT PARENT
SMG INVALID DATA LENGTH
SMG NOT BOUND
SMG RETRY SUB ALLOC
KBD DETACHED
VIO DETACHED
MOU DETACHED
VIO FONT
VIO USER FONT
VIO BAD CP
VIO NO CP
VIO NA CP
INVALID CODE PAGE
CPLIST TOO SMALL
CP NOT MOVED
MODE SWITCH INIT
CODE PAGE NOT FOUND
UNEXPECTED SLOT RETURNED
SMG INVALID TRACE OPTION
VIO INTERNAL RESOURCE
VIO SHELL INIT
SMG NO HARD ERRORS
CP SWITCH INCOMPLETE
VIO TRANSPARENT POPUP
CRITSEC OVERFLOW
CRITSEC UNDERFLOW
VIO BAD RESERVE
INVALID ADDRESS
ZERO SELECTORS REQUESTED
NOT ENOUGH SELECTORS AVA
INVALID SELECTOR
SMG INVALID PROGRAM TYPE
SMG INVALID PGM CONTROL
SMG INVALID INHERIT OPT
VIO EXTENDED SG
VIO NOT PRES MGR SG
VIO SHIELD OWNED
VIO NO MORE HANDLES
VIO SEE LOG
499 VIO ASSOCIATED DC
500 KBD NO CONSOLE
501 MOUSE NO CONSOLE
502 MOUSE INVALID HANDLE
503 SMG INVALID DEBUG PARMS
504 KBD EXTENDED SG
505 MOU EXTENDED SG
506 SMG INVALID ICON FILE
507 TRC PID NON EXISTENT
508 TRC COUNT ACTIVE
509 TRC SUSPENDED BY COUNT
510 TRC COUNT INACTIVE
511 TRC COUNT REACHED
512 NO MC TRACE
513 MC TRACE
514 TRC COUNT ZERO
515 SMG TOO MANY DDS
516 SMG INVALID NOTIFICATION
517 LF INVALID FUNCTION
518 LF NOT AVAIL
519 LF SUSPENDED
520 LF BUF TOO SMALL
521 LF BUFFER CORRUPTED
521 LF BUFFER FULL
522 LF INVALID DAEMON
522 LF INVALID RECORD
523 LF INVALID TEMPL
523 LF INVALID SERVICE
524 LF GENERAL FAILURE
525 LF INVALID ID
526 LF INVALID HANDLE
527 LF NO ID AVAIL
528 LF TEMPLATE AREA FULL
529 LF ID IN USE
530 MOU NOT INITIALIZED
531 MOUINITREAL DONE
532 DOSSUB CORRUPTED
533 MOUSE CALLER NOT SUBSYS
534 ARITHMETIC OVERFLOW
TMR NO DEVICE
TMR INVALID TIME
PVW INVALID ENTITY
PVW INVALID ENTITY TYPE
PVW INVALID SPEC
PVW INVALID RANGE TYPE
PVW INVALID COUNTER BLK
PVW INVALID TEXT BLK
PRF NOT INITIALIZED
PRF ALREADY INITIALIZED
PRF NOT STARTED
PRF ALREADY STARTED
PRF TIMER OUT OF RANGE
PRF TIMER RESET
HPFS CHKDSK NO PARM SPACE
HPFS CHKDSK NORECOGNIZE
HPFS CHKDSK NORoot FIND
HPFS CHKDSK NOFIX FS ERROR
HPFS CHKDSK CORRECT FS ERR
HPFS CHKDSK ORGAN FIX
HPFS CHKDSK RELOC BBPDATA
HPFS CHKDSK REM CORRU BLOC
HPFS CHKDSK REM CORRUP FIL
HPFS CHKDSK FIX SPACE ALLO
HPFS NOT FORMATTED DISK
HPFS CHKDSK COR ALLOC
HPFS CHKDSK SEARC UNALLOC
HPFS CHKDSK DET LOST DATA
HPFS CHKDSK PERCENT SEARC
HPFS CHKDSK LOST DATASEARC
HPFS CHKDSK CRIT NOREAD
HPFS CHKDSK DISK INUSE
HPFS CHKDSK RECOVTEMP RELOC
HPFS TOTAL DISK SPACE
HPFS DIR KBYTES
HPFS FILE KBYTES
HPFS KBYTES AVAILABLE
HPFS CHKDSK PLACE REC FILE
HPFS CHKDSK RECO DIR AS
HPFS CHKDSK PLACEED DATA
HPFS WANT MEM
HPFS GET RETURNED
HPFS SET RETURNED
HPFS BOTH RETURNED
HPFS STOP RETURNED
HPFS SETPRTYRETURNED
HPFS ALCSG RETURNED
HPFS MSEC SET
HPFS OPTIONS
HPFS POS NUM VALUE
HPFS VALUE TOO LARGE
HPFS LAZY NOT VALID
HPFS VOLUME ERROR
HPFS VOLUME DIRTY
HPFS NEW SECTOR
HPFS FORMAT PARM ERROR
HPFS CANNOT ACCESS CONFIG
HPFS RECOV FILE
HPFS CHKDSK KBYTES RESERVE
HPFS CHKDSK KBYTES IN EA
HPFS BYTEBUF SET
HPFS FORMATTING COMPLETE
HPFS WRONG VOLUME LABEL
HPFS FMAT TOO MANY DRS
VDD UNSUPPORTED ACCESS
VDD LOCK USEAGE DENIED
TIMEOUT
VDM DOWN
VDM LIMIT
VDD NOT FOUND
INVALID CALLER
PID MISMATCH
INVALID VDD HANDLE
VLPT NO SPOOLER
VCOM DEVICE BUSY
VLPT DEVICE BUSY
NESTING TOO DEEP
VDD MISSING
BIDI INVALID LENGTH
BIDI INVALID INCREMENT
673      BIDI INVALID COMBINATION
674      BIDI INVALID RESERVED
675      BIDI INVALID EFFECT
676      BIDI INVALID CSBREC
677      BIDI INVALID CSSTATE
678      BIDI INVALID LEVEL
679      BIDI INVALID TYPE SUPPORT
680      BIDI INVALID ORIENTATION
681      BIDI INVALID NUM SHAPE
682      BIDI INVALID CSD
683      BIDI NO SUPPORT
684      NO BIDI RW INCOMPLETE
689      HPFS LAZY ON
690      HPFS LAZY OFF
691      IMP INVALID PARM
692      IMP INVALID LENGTH
693      MSG HPFS DISK WARN
694      MSG HPFS FNODE WARN
730      MON BAD BUFFER
731      MODULE CORRUPTED
732      BOOT DRIVE NOT ACCESSIBLE
1477     SM OUTOF SWAPFILE
2055     LF TIMEOUT
2057     LF SUSPEND SUCCESS
2058     LF RESUME SUCCESS
2059     LF REDIRECT SUCCESS
2060     LF REDIRECT FAILURE
32768    SWAPPER NOT ACTIVE
32769    INVALID SWAPI
32770    IOERR SWAP FILE
32771    SWAP TABLE FULL
32772    SWAP FILE FULL
32773    CANT INIT SWAPPER
32774    SWAPPER ALREADY INIT
32775    PMM INSUFFICIENT MEMORY
32776    PMM INVALID FLAGS
32777    PMM INVALID ADDRESS
32778    PMM LOCK FAILED
32779    PMM UNLOCK FAILED
32780  PMM MOVE INCOMPLETE
32781  UCOM DRIVE RENAMED
32782  UCOM FILENAME TRUNCATED
32783  UCOM BUFFER LENGTH
32784  MON CHAIN HANDLE
32785  MON NOT REGISTERED
32786  SMG ALREADY TOP
32787  PMM ARENA MODIFIED
32788  SMG PRINTER OPEN
32789  PMM SET FLAGS FAILED
32790  INVALID DOS DD
32791  BLOCKED
32792  NOBLOCK
32793  INSTANCE SHARED
32794  NO OBJECT
32795  PARTIAL ATTACH
32796  INCACHE
32797  SWAP IO PROBLEMS
32798  CROSSES OBJECT BOUNDARY
32799  LONGLOCK
32800  SHORTLOCK
32801  UVIRTLOCK
32802  ALIASLOCK
32803  ALIAS
32804  NO MORE HANDLES
32805  SCAN TERMINATED
32806  TERMINATOR NOT FOUND
32807  NOT DIRECT CHILD
32808  DELAY FREE
32809  GUARDPAGE
32810  SWAPERROR
32811  LDRERROR
32812  NOMEMORY
32813  NOACCESS
32814  NO DLL TERM
65026  CPSIO CODE PAGE INVALID
65027  CPSIO NO SPOOLER
65028  CPSIO FONT ID INVALID
65033  CPSIO INTERNAL ERROR
65034  CPSIO INVALID PTR NAME
65037  CPSIO NOT ACTIVE
65039  CPSIO PID FULL
65040  CPSIO PID NOT FOUND
65043  CPSIO READ CTL SEQ
65045  CPSIO READ FNT DEF
65047  CPSIO WRITE ERROR
65048  CPSIO WRITE FULL ERROR
65049  CPSIO WRITE HANDLE BAD
65074  CPSIO SWIT LOAD
65077  CPSIO INV COMMAND
65078  CPSIO NO FONT SWIT
65079  ENTRY IS CALLGATE
0xFF00  USER DEFINED BASE

Dos INT 24 Critical Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I24 WRITE PROTECT</td>
</tr>
<tr>
<td>1</td>
<td>I24 BAD UNIT</td>
</tr>
<tr>
<td>2</td>
<td>I24 NOT READY</td>
</tr>
<tr>
<td>3</td>
<td>I24 BAD COMMAND</td>
</tr>
<tr>
<td>4</td>
<td>I24 CRC</td>
</tr>
<tr>
<td>5</td>
<td>I24 BAD LENGTH</td>
</tr>
<tr>
<td>6</td>
<td>I24 SEEK</td>
</tr>
<tr>
<td>7</td>
<td>I24 NOT DOS DISK</td>
</tr>
<tr>
<td>8</td>
<td>I24 SECTOR NOT FOUND</td>
</tr>
<tr>
<td>9</td>
<td>I24 OUT OF PAPER</td>
</tr>
<tr>
<td>10</td>
<td>I24 WRITE FAULT</td>
</tr>
<tr>
<td>11</td>
<td>I24 READ FAULT</td>
</tr>
<tr>
<td>12</td>
<td>I24 GEN FAILURE</td>
</tr>
<tr>
<td>13</td>
<td>I24 DISK CHANGE</td>
</tr>
<tr>
<td>15</td>
<td>I24 WRONG DISK</td>
</tr>
<tr>
<td>16</td>
<td>I24 UNCERTAIN MEDIA</td>
</tr>
<tr>
<td>17</td>
<td>I24 CHAR CALL INTERRUPTED</td>
</tr>
<tr>
<td>18</td>
<td>I24 NO MONITOR SUPPORT</td>
</tr>
<tr>
<td>19</td>
<td>I24 INVALID PARAMETER</td>
</tr>
<tr>
<td>20</td>
<td>I24 DEVICE IN USE</td>
</tr>
<tr>
<td>21</td>
<td>I24 QUIET INIT FAIL</td>
</tr>
</tbody>
</table>
OS/2 System Exception Codes

Exception values are 32-bit values laid out as follows:

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

where

- `Sev` - is the severity code
  - 00 - Success
  - 01 - Informational
  - 10 - Warning
  - 11 - Error

- `C` - is the Customer code flag
- `Facility` - is the facility code
- `Code` - is the facility's status code

Exceptions specific to OS/2 2.0 (e.g. XCPT_SIGNAL) will be marked with a facility code of 1.

### Examples

- **80000001H**
  - XCPT_GUARD_PAGE_VIOLATION
  - **P1**
    - **Access Code**
    - 00000001H  XCPT_READ_ACCESS
    - 00000002H  XCPT_WRITE_ACCESS
  - **P2**
    - **FaultAddr**

- **80010001H**
  - XCPT_UNABLE_TO_GROW_STACK

- **0C0010001H**
  - XCPT_PROCESS_TERMINATE

  This exception is sent to a thread for Synchronous Process Termination and also for Thread Termination.

- **0C0010002H**
  - XCPT_ASYNC_PROCESS_TERMINATE
  - **P1**
    - **TID of 'terminator' thread**

- **0C0010003H**
  - XCPT_SIGNAL
  - **P1**
    - **Signal Number**
    - 1  XCPT_SIGNAL_INTR
    - 3  XCPT_SIGNAL_KILLPROC
    - 4  XCPT_SIGNAL_BREAK

- **0C0010004H**
  - XCPT_B1NPX_ERRATA_02

- **0C0000005H**
  - XCPT_ACCESS_VIOLATION
This relates to Traps 0x09, 0x0b, 0x0c, 0x0d and 0x0e.

P1

<table>
<thead>
<tr>
<th>Access Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000H</td>
<td>XCPT_UNKNOWN_ACCESS</td>
</tr>
<tr>
<td>00000001H</td>
<td>XCPT_READ_ACCESS</td>
</tr>
<tr>
<td>00000002H</td>
<td>XCPT_WRITE_ACCESS</td>
</tr>
<tr>
<td>00000004H</td>
<td>XCPT_EXECUTE_ACCESS</td>
</tr>
<tr>
<td>00000008H</td>
<td>XCPT_SPACE_ACCESS</td>
</tr>
<tr>
<td>00000010H</td>
<td>XCPT_LIMIT_ACCESS</td>
</tr>
</tbody>
</table>

P2

<table>
<thead>
<tr>
<th>FaultAddr</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0C000006H</td>
<td>XCPT_IN_PAGE_ERROR</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x0e.</td>
</tr>
<tr>
<td>0C000001CH</td>
<td>XCPT_ILLEGAL_INSTRUCTION</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x06.</td>
</tr>
<tr>
<td>0C000001DH</td>
<td>XCPT_INVALID_LOCK_SEQUENCE</td>
</tr>
<tr>
<td>0C000024H</td>
<td>XCPT_NONCONTINUABLE_EXCEPTION</td>
</tr>
<tr>
<td>0C000025H</td>
<td>XCPT_INVALID_DISPOSITION</td>
</tr>
<tr>
<td>0C000026H</td>
<td>XCPT_UNWIND</td>
</tr>
<tr>
<td>0C000027H</td>
<td>XCPT_BAD_STACK</td>
</tr>
<tr>
<td>0C000028H</td>
<td>XCPT_INVALID_UNWIND_TARGET</td>
</tr>
<tr>
<td>0C000093H</td>
<td>XCPT_ARRAY_BOUNDS_EXCEEDED</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x05.</td>
</tr>
<tr>
<td>0C000094H</td>
<td>XCPT_FLOAT_DENORMAL_OPERAND</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x10.</td>
</tr>
<tr>
<td>0C000095H</td>
<td>XCPT_FLOAT_DIVIDE_BY_ZERO</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x10.</td>
</tr>
<tr>
<td>0C000096H</td>
<td>XCPT_FLOAT_INEXACT_RESULT</td>
</tr>
<tr>
<td></td>
<td>This relates to Trap 0x10.</td>
</tr>
<tr>
<td>0C000097H</td>
<td>XCPT_FLOAT_INVALID_OPERATION</td>
</tr>
</tbody>
</table>
This relates to Trap 0x10.

0C0000098H
XCPT_FLOAT_OVERFLOW
This relates to Trap 0x10.

0C0000099H
XCPT_FLOAT_STACK_CHECK
This relates to Trap 0x10.

0C000009AH
XCPT_FLOAT_UNDERFLOW
This relates to Trap 0x10.

0C000009BH
XCPT_INTEGER_DIVIDE_BY_ZERO
This relates to Trap 0x00.

0C000009CH
XCPT_INTEGER_OVERFLOW
This relates to Trap 0x04.

0C000009DH
XCPT_PRIVILEGED_INSTRUCTION
This relates to Trap 0x0d.

0C000009EH
XCPT_DATATYPE_MISALIGNMENT
This relates to Trap 0x11.

P1  Access Code
  00000001H  XCPT_READ_ACCESS
  00000002H  XCPT_WRITE_ACCESS

P2  Alignment

P3  FaultAddr

0C000009FH
XCPT_BREAKPOINT
This relates to Trap 0x03.

0C00000AH
XCPT_SINGLE_STEP
This relates to Trap 0x01.

For a further information refer to:

- OS/2 Technical Library - Control Program Programming Reference, Appendix C.
- bsexcpt.h or bsexcpt.inc include files supplied with the OS/2 Programmers Toolkit.

-------------------------------------------------------------

Trap and Exeption Popup Message Reference

The trap screen and has in two basic formats:
The application exception (SYS0147, SYS317x, SYS3190) messages.

The Internal Processing Error (IPE).

**Application Trap/Exception**

Application exception popups are logged in the POPUPLOG.OS2 file of which the following is an example. They are also displayed in a popup window in a slightly abbreviated form.

Control of exception logging and popup displays may be done from the TRAPLOG command or SUPPRESSPOPUPS CONFIG.SYS statement.

01-> 02-25-1999  10:58:35  SYS3175  PID 00b2  TID 0001  Slot 0068
02-> E:\CLASSES\LABS\LAB26\BEDBUG.EXE
03-> c0000005
04-> 1bf94e24
05-> P1=00000001  P2=00000000  P3=XXXXXXXX  P4=XXXXXXXX
06-> EAX=00000000  EBX=00060210  ECX=0002881c  EDX=00060210
07-> EDI=00000002  EDI=00000002
08-> DS=0053  DSACC=d0f3  DSLIM=1fffffff
09-> ES=0053  ESACC=d0f3  ESLIM=1fffffff
10-> FS=150b  FSACC=00f3  FSLIM=00000030
11-> GS=0000  GSACC=****  GSLIM=********
12-> CS:EIP=005b:1bf94e24  CSACC=d0df  CSLIM=1fffffff
13-> SS:ESP=0053:000287ec  SSACC=d0f3  SSLIM=1fffffff
14-> EBP=000287f8  FLG=00012206
15-> DOSCALL1.DLL 0002:00004e24

The information presented varies slightly according to circumstance. In general, inapplicable information is either omitted, or overlayed with asterisks (*) or exes (X).

Each line of the trap screen conveys the following meaning:

1. Date and Time or Trap, Trap message Id and Failing Process Id, Thread Id and Thread Slot.
2. Failing process. In general this will not be the trapping module.
3. OS/2 System Exception code. See OS/2 System Exception Codes for a complete set of system generated exceptions.
4. 32-bit Instruction address at time of exception.
5. Exception Information Parameters. taken from the Exception Report Record. See OS/2 System Exception Codes for the exception information parameters that are associated with each system exception.

**Note:**

When P1=00000000 and P2=FFFFFFFF this frequently indicates that the trap occurred while executing a system API and that the previous instruction in the user’s code was a call gate. When this happen a bad parameter has been passed to the system from the application.

If in addition the EFLAGS register has high word 0019xxxx then Virtual Machine Extensions may indirectly be causing a problem. Try VME=NO in CONFIG.SYS.

6. The EAX, EXB, EXC and EDX registers at the time the exception was reported.
7. The ESI and EDI registers at the time the exception was reported.
8. The DS selector at the time the exception was reported.

This information is presented in the form:

xS=nnnn The selector value.
exSACC=nnnn The descriptor access bits.

Reading from right to left the bits of the access field are assigned the following meaning:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(A) 1=Accessed</td>
</tr>
<tr>
<td>1</td>
<td>(W) 1=Writable</td>
</tr>
<tr>
<td>2</td>
<td>(E) 1=Executable</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
4 (S) 1=Application 0=System
5 & 6 (DPL) Privilege Level
7 (P) 1=Segment present
8 - 11 0
12 (AVL) 1=UVIRT allocation
13 (D) 1=32-bit Operands/Data
14 0
15 (G) 1=4K granularity limit, 0=byte granularity limit

See the "INTEL Pentium User's Guide, Volume 3" for more information on descriptor formats.

xSLIM=nnnnnnnn
The limit field from the descriptor.

9. The ES selector at the time the exception was reported.
10. The FS selector at the time the exception was reported.
11. The GS selector at the time the exception was reported.
12. The instruction address at the time the exception was reported, followed by the CS selector Limit and Access fields.
13. The stack address at the time the exception was reported, followed by the SS selector Limit and Access fields.
14. The EBP register and EFLAGS register.
15. The module name and relative object and offset within the module that corresponds to the exception address reported on line 4.

Notes:

Lines 3-5 are formatted from the Exception Report Record generated at the time the exception occurred.

Lines 6-14 are formatted from the Exception Context Record generated at the time the exception occurred.

Exception report and Context records can be modified by exception handlers so it is possible that the information displayed might not be correct.

Exception popups will not be generated if an exception handler attempts recovery by returning XCPT_CONTINUE_EXECUTION to the system. However this does not guarantee that the program will continue to operate correctly. When recovery is not successful it might be necessary to disable exception handlers. For Presentation Manager and the Workplace Shell this can be done by specifying SET SHAPIEXCEPTIONHANDLER=OFF and SHELLEXCEPTIONHANDLER=OFF in CONFIG.SYS. This can also be done from the kernel debugger by setting the first double-word of a thread's TIB to 0xffffffff. An alternative approach is to use system trace to log exception handler dispatching. Minor codes 361, 362 and 363 of DOSCALL1 will log exception report and context records before and after they have been processed by and exception handler. Minor code 361 logs the original exception information, while minor code 363 logs the exception information that is returned to the system (and used for the popup log). Minor code 262 logs exception information before each exception handler is called. The Exception Registration Record is also logged by the tracepoint.

To activate these tracepoints issue the following command sequence:

```
TRACE ON /B:512 /D:PROCNAME,TID
TRACE ON DOSCALL1(361,362,363)
```

The first command defines a system trace buffer of 512Kb and turns on logging of Process name and Thread Id with each trace record. The second command activates the tracepoints. Use the TRACEGET and/or the TRACEMFT commands to extract and view the traced events.

Using trace to log exceptions will also provide full trap information for floating point exception, which is not provided by any other means.

The exception address reported in line 4 in most cases agrees with the CS:EIP reported in line 12. However, when a nested exception occurs, the register information will relate to the most recent exception. A particular example of this is where an Exit List Handler traps. Exit List Handlers are called when a process terminates, after any exception handling. The system first generates an XCPT_PROCESS_TERMINATE exception, which may be handled by exception handlers. If this exception is not recovered then process termination continues with Exit List processing. Once this starts, application exception handlers will not be called if any further exceptions are generated. If a further exception is generated then it will become a nested exception of the original XCPT_PROCESS_TERMINATE. If an Exit List Handler traps, a SYS3170 popup will be generated with the register information in lines 6 - 14 corresponding to the nested exception and lines 3 - 5 and 15 corresponding to the original XCPT_PROCESS_TERMINATE exception.

If the system is unable to generate an exception popup message then a SYS0147 is generated. This can happen when there is insufficient kernel heap memory left to allocate a HARDERR request packet. This is not the only cause of a SYS0147, PM...
resource (heap) shortages also cause this message. is exhausted this can occur.

SYS3190 occurs because of a TRAP 6. The application is incorrectly using LOCK prefixes either deliberately or possibly it had taken a jump to a non-instruction boundary.

**System Internal Processing Error (IPE)**

The IPE message appears because of a fatal internal error condition. This may or may not be a trap, though the IPE trap is the most common.

The IPE message has the general format:

1. <IPE specific Message>
2. THE SYSTEM DETECTED AN INTERNAL PROCESSING ERROR AT LOCATION ##xxxx:yyyyyyyy - aaaa:bbbb
3. 111111, ffff
4. 038600d1
5. INTERNAL REVISION 6.307 DATE: 92/03/01

The parts of the IPE message are:

1. **IPE specific message**, which could be a simple line of text, for example:
   
   CPS: Empty allocation block--not supported.

   or a formatted register dump for a system trap, such as:

   TRAP 0002  ERRCD= 0000  ERACC= ****  ERLIM= ********
   EAX= 7d240a58  EBX= ff202fde  ECX= 00064423  EDX= 0003624
   ES= fff32772c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
   CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
   SS:EIP= 0030 : 00004a38  SSACC= 1097  SSLIM= 0003fff
   DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= ffffffff
   ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
   FS= 0000  FSACC= ****  FSLIM= ********
   GS= 0000  GSACC= ****  GSLIM= ********

   THE SYSTEM DETECTED AN INTERNAL PROCESSING

2. The CS:EIP of the caller to the kernel panic routine is shown as ##xxxx:yyyyyyyy. For traps this will always be an address within the trap handler and not the address at which the error occurred - that is given in the error specific message.

   The CS:EIP is prefixed with either ## to indicate protect mode, paging enables in accordance with the Kernel Debugger command prompt.

   The kernel relative object:offset address is shown as aaaa:bbbb.

3. ### is intended to be the source line number at which the panic occurred. Values greater than 60000 are generated panic sequence numbers, ###### is the source file number.

   **Note:** These values are mostly arbitrary and therefore not particularly useful. There is no published cross reference.

4. The processor ID.

5. The kernel revision information.

An example of the IPE trap screen is show in the following diagram:

1-> TRAP 0002  ERRCD= 0000  ERACC= ****  ERLIM= ********
EAX= 7d240a58  EBX= ff202fde  ECX= 00064423  EDX= 0003624
ES= fff32772c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
SS:EIP= 0030 : 00004a38  SSACC= 1097  SSLIM= 0003fff
DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= ffffffff
ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
FS= 0000  FSACC= ****  FSLIM= ********
GS= 0000  GSACC= ****  GSLIM= ********
THE SYSTEM DETECTED AN INTERNAL PROCESSING
The register information may be interpreted as for application trap screens, with the following points notes:

1. This line shows the Trap number followed by the INTEL error code. Most often the associated error code is a selector number. When this is the case, this line formats the selector's access and limit values.

2. This line shows the address at which the trap occurred.

3. The value of control register 0 (CR0) is formatted after the DS register.
   CR0 contains processor control mode settings.

4. The value of control register 2 (CR2) is formatted after the ES register.
   CR2 contains the fault address for TRAP E errors.

--------------------------------------------

NMI Error Codes

NMI exceptions (TRAP 2) have no hardware defined Error Code. OS/2 uses the error code in the trap screen to indicate the source of the NMI as follows:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Msg No</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>SYS1944</td>
<td>Software caused NMI (INT 2)</td>
</tr>
<tr>
<td>0001</td>
<td>SYS1945</td>
<td>RAM error, check memory (parity error)</td>
</tr>
<tr>
<td>0002</td>
<td>SYS1946</td>
<td>Adapter caused error (I/O channel check)</td>
</tr>
<tr>
<td>0003</td>
<td>SYS1947</td>
<td>Check bus mastering adapters, update adapter drivers. Also disable bus mastering on failing adapters as a problem determination tool to figure out which adapter is causing the failure. Contact the adapter vendor for further assistance. (DMA timeout)</td>
</tr>
<tr>
<td>0004</td>
<td>SYS1948</td>
<td>A device driver or Dos application disabled interrupts too long. Contact appropriate software vendor for updated software (Watchdog timeout).</td>
</tr>
<tr>
<td>0005</td>
<td>SYS3140</td>
<td>Contact application or device driver hardware vendor (software generated NMI).</td>
</tr>
<tr>
<td>0006</td>
<td>SYS3141</td>
<td>see error code 0003.</td>
</tr>
<tr>
<td>0007</td>
<td>SYS3142</td>
<td>see error code 0004.</td>
</tr>
<tr>
<td>0008</td>
<td>SYS3143</td>
<td>see error code 0002.</td>
</tr>
<tr>
<td>0009</td>
<td>SYS3149</td>
<td>see error code 0001.</td>
</tr>
</tbody>
</table>

--------------------------------------------

Standard GDT Assignments
The following table lists the GDT assignments that are statically assigned or assigned dynamically during initialisation.

This list is subject to change from release to release but may be verified by listing symbols from OS2KRNL segment DOSGDTDATA using the Kernel Debugger LS command.

Note:

The Callgate descriptor assignments are shown for the ALLSTRCT kernel. For the RETAIL kernel they begin at one GTD entry earlier. Thus GTD_DOSALLOCSEG through GTD_R0CSC are assigned to selectors 1d08 through 1ea0.

<table>
<thead>
<tr>
<th>Selector</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GDT</td>
<td>entry 0 is reserved (invalid)</td>
</tr>
<tr>
<td>8</td>
<td>GDT_GDT</td>
<td>entry 8 used to be GDT (now invalid)</td>
</tr>
<tr>
<td>10</td>
<td>GDT_TSS</td>
<td>Protect mode TSS</td>
</tr>
<tr>
<td>18</td>
<td>GDT_IDT</td>
<td>Protect Mode IDT</td>
</tr>
<tr>
<td>20</td>
<td>GDT_RM_IDT</td>
<td>Selector for 1st 1K</td>
</tr>
<tr>
<td>28</td>
<td>GDT_LDT</td>
<td>Selector for LDT</td>
</tr>
<tr>
<td>30</td>
<td>GDT_PTDA</td>
<td>PTDA/TCB/TSD selector</td>
</tr>
<tr>
<td>38</td>
<td>GDT_FPEM</td>
<td>Floating Point Emulator Work Area</td>
</tr>
<tr>
<td>40</td>
<td>GDT_ROMDATA</td>
<td>ROM data at physical address 400h</td>
</tr>
<tr>
<td>4a</td>
<td>GDT_R2DS</td>
<td>Ring 2 Data Selector</td>
</tr>
<tr>
<td>53</td>
<td>GDT_R3DS</td>
<td>Ring 3 Data Selector</td>
</tr>
<tr>
<td>5b</td>
<td>GDT_R3CS</td>
<td>Ring 3 Code Selector</td>
</tr>
<tr>
<td>63</td>
<td>GDT_R3PDS</td>
<td>Ring 3 Protected Data Selector</td>
</tr>
<tr>
<td>6b</td>
<td>GDT_R3THKDS,</td>
<td>Ring 3 Thunk Data Selector</td>
</tr>
<tr>
<td>70</td>
<td>GDT_SAS,</td>
<td>System Anchor Segment</td>
</tr>
<tr>
<td>78</td>
<td>GDT_DOSALIAS</td>
<td>SAS Read/Write Alias</td>
</tr>
<tr>
<td>80</td>
<td>GDT_SYSINFOSEG</td>
<td>InfosegGDT</td>
</tr>
<tr>
<td>88</td>
<td>GDT_DFTSS</td>
<td>Double Fault TSS</td>
</tr>
<tr>
<td>90</td>
<td>GDT_DFSTACK</td>
<td>Trap 8 stack selector</td>
</tr>
<tr>
<td>98</td>
<td>GDT_VPB</td>
<td>VPB BMP Segment</td>
</tr>
<tr>
<td>a0</td>
<td>GDT_RDR1</td>
<td>Reserved</td>
</tr>
<tr>
<td>a8</td>
<td>GDT_Buffers</td>
<td>Buffer Pool Segment</td>
</tr>
<tr>
<td>b0</td>
<td>GDT_Unused</td>
<td>unused selector (used to be MFT)</td>
</tr>
<tr>
<td>b8</td>
<td>GDT_RLR</td>
<td>RLR selector</td>
</tr>
<tr>
<td>c0</td>
<td>GDT_SFT</td>
<td>SFT selector of first SFT segment</td>
</tr>
<tr>
<td>c8</td>
<td>GDT_FSC</td>
<td>FSC array segment selector</td>
</tr>
<tr>
<td>d0</td>
<td>GDT_mFSD</td>
<td>mini-FSD</td>
</tr>
<tr>
<td>d8</td>
<td>GDT_RIPL</td>
<td>Remote IPL data</td>
</tr>
<tr>
<td>e0</td>
<td>GDT_NULLIDT</td>
<td>Invalid descriptor for mode switch</td>
</tr>
<tr>
<td>e8</td>
<td>GDT_INTSTACK</td>
<td>Interrupt stack alias</td>
</tr>
<tr>
<td>Address</td>
<td>Register</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>f0</td>
<td>GDT_RMCODE</td>
<td>386 modesw code selector</td>
</tr>
<tr>
<td>f8</td>
<td>GDT_RMDATA</td>
<td>386 modesw data selector</td>
</tr>
<tr>
<td>100</td>
<td>DOSHLP_CODESEL</td>
<td>DosHlp Code Selector</td>
</tr>
<tr>
<td>108</td>
<td>GDT_Pool</td>
<td>Start of dynamic GDT allocations</td>
</tr>
<tr>
<td>1508</td>
<td>GDT_Poolend</td>
<td>End of dynamic GDT allocations</td>
</tr>
<tr>
<td>150b</td>
<td>GDT_TIB</td>
<td>TIB selector</td>
</tr>
<tr>
<td>1d10</td>
<td>GDT_DOSALLOCSEG</td>
<td>DOSALLOCSEG call gate</td>
</tr>
<tr>
<td>1d18</td>
<td>GDT_DOSALLOCPROTSEG</td>
<td>DOSALLOCPROTSEG call gate</td>
</tr>
<tr>
<td>1d20</td>
<td>GDT_DOSDYNAMICTRACE</td>
<td>DOSDYNAMICTRACE call gate</td>
</tr>
<tr>
<td>1d28</td>
<td>GDT_DOSError</td>
<td>DOSError call gate</td>
</tr>
<tr>
<td>1d30</td>
<td>GDT_DOSFREEERESOURCE</td>
<td>DOSFREEERESOURCE call gate</td>
</tr>
<tr>
<td>1d38</td>
<td>GDT_DOSQUERYABIOSUPPORT</td>
<td>DOSQUERYABIOSUPPORT call gate</td>
</tr>
<tr>
<td>1d40</td>
<td>GDT_DOS16LDRDIRTYWORKER</td>
<td>DOS16LDRDIRTYWORKER call gate</td>
</tr>
<tr>
<td>1d48</td>
<td>GDT_DOSFREESEG</td>
<td>DOSFREESEG call gate</td>
</tr>
<tr>
<td>1d50</td>
<td>GDT_DOSGETPROCADDR</td>
<td>DOSGETPROCADDR call gate</td>
</tr>
<tr>
<td>1d58</td>
<td>GDT_DOSIXECPGM</td>
<td>DOSIXECPGM call gate</td>
</tr>
<tr>
<td>1d60</td>
<td>GDT_DOSIQAPPTYPE</td>
<td>DOSIQAPPTYPE call gate</td>
</tr>
<tr>
<td>1d68</td>
<td>GDT_DOSISEMWAIT</td>
<td>DOSISEMWAIT call gate</td>
</tr>
<tr>
<td>1d70</td>
<td>GDT_DOSLOADMODULE</td>
<td>DOSLOADMODULE call gate</td>
</tr>
<tr>
<td>1d78</td>
<td>GDT_DOSMAKEPIPE</td>
<td>DOSMAKEPIPE call gate</td>
</tr>
<tr>
<td>1d80</td>
<td>GDT_DOSREALLOCSEG</td>
<td>DOSREALLOCSEG call gate</td>
</tr>
<tr>
<td>1d88</td>
<td>GDT_DOSICG</td>
<td>DOSICG call gate</td>
</tr>
<tr>
<td>1d90</td>
<td>GDT_PANICWRITE</td>
<td>PANICWRITE call gate</td>
</tr>
<tr>
<td>1d98</td>
<td>GDT_DOSSETPRTY</td>
<td>DOSSETPRTY call gate</td>
</tr>
<tr>
<td>1da0</td>
<td>GDT_DOSLOGMODE</td>
<td>DOSLOGMODE call gate</td>
</tr>
<tr>
<td>1da8</td>
<td>GDT_DOSSETCP</td>
<td>DOSSETCP call gate</td>
</tr>
<tr>
<td>1db0</td>
<td>GDT_DOSGLOBALSEG</td>
<td>DOSGLOBALSEG call gate</td>
</tr>
<tr>
<td>1db8</td>
<td>GDT_DOSCREATETHREAD</td>
<td>DOSCREATETHREAD call gate</td>
</tr>
<tr>
<td>1dc0</td>
<td>GDT_DOSEXIT</td>
<td>DOSEXIT call gate</td>
</tr>
<tr>
<td>1dc8</td>
<td>GDT_DOSEXITLIST</td>
<td>DOSEXITLIST call gate</td>
</tr>
<tr>
<td>1dd0</td>
<td>GDT_DOSFREEMODULE</td>
<td>DOSFREEMODULE call gate</td>
</tr>
<tr>
<td>1dd8</td>
<td>GDT_DOSRESUMETHREAD</td>
<td>DOSRESUMETHREAD call gate</td>
</tr>
<tr>
<td>1de0</td>
<td>GDT_DOSSLEEP</td>
<td>DOSSLEEP call gate</td>
</tr>
<tr>
<td>1de8</td>
<td>GDT_DOSSUSPENDTHREAD</td>
<td>DOSSUSPENDTHREAD call gate</td>
</tr>
<tr>
<td>1df0</td>
<td>GDT_DOSLIBINIT</td>
<td>DOSLIBINIT call gate</td>
</tr>
<tr>
<td>1df8</td>
<td>GDT_REDIR</td>
<td>REDIR call gate</td>
</tr>
<tr>
<td>1e00</td>
<td>GDT_DOSCHGFILEPTR</td>
<td>DOSCHGFILEPTR call gate</td>
</tr>
<tr>
<td>1e08</td>
<td>GDT_DOSPROTECTCHGFILEPTR</td>
<td>DOSPROTECTCHGFILEPTR call gate</td>
</tr>
<tr>
<td>1e10</td>
<td>GDT_DOSCLOSE</td>
<td>DOSCLOSE call gate</td>
</tr>
<tr>
<td>1e18</td>
<td>GDT_DOSPROTECTCLOSE</td>
<td>DOSPROTECTCLOSE call gate</td>
</tr>
</tbody>
</table>
Standard LDT Assignments

The following table lists the LDT assignments that are defined by the system.

<table>
<thead>
<tr>
<th>Selector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Read/Only access to the current LDT</td>
</tr>
<tr>
<td>dff7</td>
<td>Read/Only access to the current Global Information Segment.</td>
</tr>
<tr>
<td>dfff</td>
<td>Read/Only access to the current Local Information Segment and the current thread's Thread Local Memory Area.</td>
</tr>
</tbody>
</table>

VM System Object Owner IDs

*System Object Owner identifiers* are a reserved range of *hobs* used as labels for identifying generic types of memory object. They are not handles to *VMOBs*, but are used in the *own* and *hmtie* fields of some *VMOBs* and also in the *own* fields of some *Kernel Heap Block* headers.

*Note:* From Warp 3.0 fix pack 35 and Warp 4.0 GA, system object ids for file system and physical device driver owners are no longer used.
Instead the true module handle of the driver module is used.

The following table lists the system objects IDs are defined. The names shown are those displayed by the Kernel Debugger and Dump Formatter when formatting VMOB structures:

<table>
<thead>
<tr>
<th>Name</th>
<th>IL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lielist</td>
<td>0xff2d</td>
<td>LDR LieLists</td>
</tr>
<tr>
<td>demversion</td>
<td>0xff2e</td>
<td>DEM fake version entries</td>
</tr>
<tr>
<td>vmbmapd</td>
<td>0xff2f</td>
<td>VM Arena Bitmap Directory</td>
</tr>
<tr>
<td>npipenpn</td>
<td>0xff30</td>
<td>Named pipe NPN segment</td>
</tr>
<tr>
<td>npipenp</td>
<td>0xff31</td>
<td>Named pipe NP segment</td>
</tr>
<tr>
<td>reqpktttcb</td>
<td>0xff32</td>
<td>DD TCB request packets</td>
</tr>
<tr>
<td>reqpkt2</td>
<td>0xff33</td>
<td>DD strat2 request packets</td>
</tr>
<tr>
<td>spldevrmp</td>
<td>0xff34</td>
<td>Spool Dev RMP segment</td>
</tr>
<tr>
<td>chardevrmp</td>
<td>0xff35</td>
<td>Char Dev RMP segment</td>
</tr>
<tr>
<td>syssemrmp</td>
<td>0xff36</td>
<td>System Semaphore RMP segment</td>
</tr>
<tr>
<td>romdata</td>
<td>0xff37</td>
<td>ROM data</td>
</tr>
<tr>
<td>libpath</td>
<td>0xff38</td>
<td>LDR LibPath</td>
</tr>
<tr>
<td>jfnflags</td>
<td>0xff39</td>
<td>JFN flags</td>
</tr>
<tr>
<td>jfntable</td>
<td>0xff3a</td>
<td>JFN table</td>
</tr>
<tr>
<td>ptouvirt</td>
<td>0xff3b</td>
<td>PhysToUVirt</td>
</tr>
<tr>
<td>tkr3stack</td>
<td>0xff3c</td>
<td>Ring 3 stack</td>
</tr>
<tr>
<td>tkr2stack</td>
<td>0xff3d</td>
<td>Ring 2 stack</td>
</tr>
<tr>
<td>tkenv</td>
<td>0xff3e</td>
<td>User Environment</td>
</tr>
<tr>
<td>tktib</td>
<td>0xff3f</td>
<td>Thread Information Block</td>
</tr>
<tr>
<td>reqpkt1</td>
<td>0xff40</td>
<td>DD strat1 request packets</td>
</tr>
<tr>
<td>allocphys</td>
<td>0xff41</td>
<td>Allocated via DevHlp AllocPhys</td>
</tr>
<tr>
<td>khbdon</td>
<td>0xff42</td>
<td>Unusable donated heap page owner</td>
</tr>
<tr>
<td>krhrwlm</td>
<td>0xff43</td>
<td>Resident R/W 1Meg mem heap owner</td>
</tr>
<tr>
<td>krholm</td>
<td>0xff44</td>
<td>Resident R/W 1Meg mem heap owner</td>
</tr>
<tr>
<td>mmph</td>
<td>0xff45</td>
<td>dekko mapped memory</td>
</tr>
<tr>
<td>pageio</td>
<td>0xff46</td>
<td>pageio per-swap-file save block</td>
</tr>
<tr>
<td>fsreclok</td>
<td>0xff47</td>
<td>record lock record owner</td>
</tr>
</tbody>
</table>

**File System Drivers**

<table>
<thead>
<tr>
<th>Name</th>
<th>IL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsd1</td>
<td>0xff48</td>
<td>PSD 1</td>
</tr>
<tr>
<td>fsd2</td>
<td>0xff49</td>
<td>PSD 2</td>
</tr>
<tr>
<td>fsd3</td>
<td>0xff4a</td>
<td>PSD 3</td>
</tr>
<tr>
<td>fsd4</td>
<td>0xff4b</td>
<td>PSD 4</td>
</tr>
<tr>
<td>fsd5</td>
<td>0xff4c</td>
<td>PSD 5</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------</td>
<td></td>
</tr>
<tr>
<td>fsd6</td>
<td>0xff4d FSD 6</td>
<td></td>
</tr>
<tr>
<td>fsd7</td>
<td>0xff4e FSD 7</td>
<td></td>
</tr>
<tr>
<td>fsd8</td>
<td>0xff4f FSD 8 and subsequent</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Device Drivers</strong></td>
<td></td>
</tr>
<tr>
<td>dd1</td>
<td>0xff50 device driver 1</td>
<td></td>
</tr>
<tr>
<td>dd2</td>
<td>0xff51 device driver 2</td>
<td></td>
</tr>
<tr>
<td>dd3</td>
<td>0xff52 device driver 3</td>
<td></td>
</tr>
<tr>
<td>dd4</td>
<td>0xff53 device driver 4</td>
<td></td>
</tr>
<tr>
<td>dd5</td>
<td>0xff54 device driver 5</td>
<td></td>
</tr>
<tr>
<td>dd6</td>
<td>0xff55 device driver 6</td>
<td></td>
</tr>
<tr>
<td>dd7</td>
<td>0xff56 device driver 7</td>
<td></td>
</tr>
<tr>
<td>dd8</td>
<td>0xff57 device driver 8</td>
<td></td>
</tr>
<tr>
<td>dd9</td>
<td>0xff58 device driver 9</td>
<td></td>
</tr>
<tr>
<td>dd10</td>
<td>0xff59 device driver 10</td>
<td></td>
</tr>
<tr>
<td>dd11</td>
<td>0xff5a device driver 11</td>
<td></td>
</tr>
<tr>
<td>dd12</td>
<td>0xff5b device driver 12</td>
<td></td>
</tr>
<tr>
<td>dd13</td>
<td>0xff5c device driver 13</td>
<td></td>
</tr>
<tr>
<td>dd14</td>
<td>0xff5d device driver 14</td>
<td></td>
</tr>
<tr>
<td>dd15</td>
<td>0xff5e device driver 15</td>
<td></td>
</tr>
<tr>
<td>dd16</td>
<td>0xff5f device driver 16 and subsequent</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Miscellaneous Owners</strong></td>
<td></td>
</tr>
<tr>
<td>fsclmap</td>
<td>0xff60 cluster map owner</td>
<td></td>
</tr>
<tr>
<td>cdsrcmp</td>
<td>0xff61 Current Directory Structure RMP seg</td>
<td></td>
</tr>
<tr>
<td>tom</td>
<td>0xff62 Timeout Manager</td>
<td></td>
</tr>
<tr>
<td>abios</td>
<td>0xff63 Advanced BIOS</td>
<td></td>
</tr>
<tr>
<td>cache</td>
<td>0xff64 cache</td>
<td></td>
</tr>
<tr>
<td>dbgdcb</td>
<td>0xff65 DBG Debug Control Block</td>
<td></td>
</tr>
<tr>
<td>dbgkdb</td>
<td>0xff66 DBG Kernel Debug Block</td>
<td></td>
</tr>
<tr>
<td>dbgwpcb</td>
<td>0xff67 DBG Watch Point Control Block</td>
<td></td>
</tr>
<tr>
<td>demsft</td>
<td>0xff68 DEM SFT array (for FCBs)</td>
<td></td>
</tr>
<tr>
<td>demfont</td>
<td>0xff69 DEM font offsets</td>
<td></td>
</tr>
<tr>
<td>demfont</td>
<td>0xff6a DEM font data</td>
<td></td>
</tr>
<tr>
<td>devhlp</td>
<td>0xff6b allocated via devhlp AllocPhys</td>
<td></td>
</tr>
<tr>
<td>discard</td>
<td>0xff6c discardable, zero fill object</td>
<td></td>
</tr>
<tr>
<td>doslhp</td>
<td>0xff6d DosHelp segment</td>
<td></td>
</tr>
<tr>
<td>dyndtp</td>
<td>0xff6e DYN tracepoint parm block</td>
<td></td>
</tr>
<tr>
<td>dyndto</td>
<td>0xff6f dynamic tracepoint</td>
<td></td>
</tr>
<tr>
<td>dyndtst</td>
<td>0xff70 tmp dynamic trace info</td>
<td></td>
</tr>
</tbody>
</table>
dynmtel 0xff71  DYN MTE dynamic trace link
emalloc 0xff72  EM86 malloc()
emtss 0xff73  EM86 TSS
device 0xff74  installed device driver
infoseg 0xff75  infoseg (local or global)
initmsg 0xff76  INIT saved message
init 0xff77  generic init-time only
intdirq 0xff78  INT IRQ info
intstack 0xff79  interrupt stack
ioplist 0xff7a  List of modules with IOPL
kdbalias 0xff7b  Kernel debugger alias
kdbsym 0xff7c  Kernel debugger symbol
kmhook 0xff7d  KM hook info
ksem 0xff7e  KSEM semaphore
lbdd 0xff7f  loadable base device driver
lid 0xff80  ABIOS logical identifier
monitor 0xff81  monitor segment
mshare 0xff82  named-shared
mshrmp 0xff83  RMP having mshare records
nmi 0xff84  non maskable interrupt
npx 0xff85  287/387 save area
orphan 0xff86  orphaned segment
prof 0xff87  profile support
ptogdt 0xff88  Allocated via dh_allocateGDTSelector
ptovirt 0xff89  PhysToVirt
puse 0xff8a  Page Usage
pusetmp 0xff8b  tmp Page Usage
perfview 0xff8c  Perfview
qscache 0xff8d  QuerySysInfo cache
ras 0xff8e  RAS segment
resource 0xff8f  Resource BMP segment
sysserv 0xff90  system service
timer 0xff91  timer services segment
traphe 0xff92  TRAP Hard Error
File System Owners
fsbuf 0xff93  file system buffer
cdevtmp 0xff94  Char DEV TMP
fsc 0xff95  FSC segment
dpb 0xff96  DPB
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eatmp</td>
<td>0xff97</td>
<td>fat EA TMP</td>
</tr>
<tr>
<td>fatsrch</td>
<td>0xff98</td>
<td>fat search segment</td>
</tr>
<tr>
<td>gnotify</td>
<td>0xff99</td>
<td>FindNotify global segment</td>
</tr>
<tr>
<td>pnotify</td>
<td>0xff9a</td>
<td>FindNotify private segment</td>
</tr>
<tr>
<td>fsh</td>
<td>0xff9b</td>
<td>installable file sys helper</td>
</tr>
<tr>
<td>ifs</td>
<td>0xff9c</td>
<td>installable file system</td>
</tr>
<tr>
<td>mfsd</td>
<td>0xff9d</td>
<td>mini file system</td>
</tr>
<tr>
<td>mft</td>
<td>0xff9e</td>
<td>master file table</td>
</tr>
<tr>
<td>npipebuf</td>
<td>0xff9f</td>
<td>Named pipe I/O buffer segment</td>
</tr>
<tr>
<td>pipe</td>
<td>0xffa0</td>
<td>pipe</td>
</tr>
<tr>
<td>sft</td>
<td>0xffa1</td>
<td>system file table</td>
</tr>
<tr>
<td>vpblk</td>
<td>0xffa2</td>
<td>volume parameter block</td>
</tr>
</tbody>
</table>

**Loader Owners**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldcache</td>
<td>0xffa3</td>
<td>Loader Instance Data Cache</td>
</tr>
<tr>
<td>ldrlld</td>
<td>0xffa4</td>
<td>LDR Dynamic Load record</td>
</tr>
<tr>
<td>invalid</td>
<td>0xffa5</td>
<td>Cache being made</td>
</tr>
<tr>
<td>ldrmte</td>
<td>0xffa6</td>
<td>mte</td>
</tr>
<tr>
<td>ldrpath</td>
<td>0xffa7</td>
<td>LDR MTE path</td>
</tr>
<tr>
<td>ldrnres</td>
<td>0xffa8</td>
<td>LDR non-resident names</td>
</tr>
<tr>
<td>prot16</td>
<td>0xffa9</td>
<td>Protect 16 list</td>
</tr>
</tbody>
</table>

**Boot Loader and Kernel Owners**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os2krnl</td>
<td>0xffa0</td>
<td>os2krnl load image</td>
</tr>
<tr>
<td>os2ldr</td>
<td>0xffa1</td>
<td>os2ldr load image</td>
</tr>
<tr>
<td>ripl</td>
<td>0xffa2</td>
<td>Remote IPL (remote boot)</td>
</tr>
</tbody>
</table>

**Page Manager Owners**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgalias</td>
<td>0xffa3</td>
<td>Temporary page manager aliases</td>
</tr>
<tr>
<td>pgbuf</td>
<td>0xffa4</td>
<td>PG loader and swapper buffer</td>
</tr>
<tr>
<td>pgcrpte</td>
<td>0xffa5</td>
<td>PG Compat. region page table</td>
</tr>
<tr>
<td>dbgalias</td>
<td>0xffa6</td>
<td>debugger alias pte</td>
</tr>
<tr>
<td>pdiag</td>
<td>0xffa7</td>
<td>PG Page directory</td>
</tr>
<tr>
<td>pgkstack</td>
<td>0xffa8</td>
<td>kernel stack region</td>
</tr>
<tr>
<td>pgvp</td>
<td>0xffa9</td>
<td>VP array</td>
</tr>
<tr>
<td>pgpf</td>
<td>0xffa0</td>
<td>PF array</td>
</tr>
<tr>
<td>pgprf</td>
<td>0xffa1</td>
<td>Page Range Table</td>
</tr>
<tr>
<td>pgsyspte</td>
<td>0xffa2</td>
<td>PG System page tables</td>
</tr>
</tbody>
</table>

**Selector Manager Owners**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdt</td>
<td>0xffa3</td>
<td>SEL GDT</td>
</tr>
<tr>
<td>selheap</td>
<td>0xffa4</td>
<td>Selector-mapped heap block</td>
</tr>
<tr>
<td>ldt</td>
<td>0xffa5</td>
<td>SEL LDT</td>
</tr>
<tr>
<td>lock</td>
<td>0xffa6</td>
<td>SEL Lock</td>
</tr>
</tbody>
</table>
**NO-OP Locks**

**SEL UVIRL mapping**

**SEM Miscellaneous**

**SEM Mux Queue**

**SEM Open Queue**

**SEM SemRecord**

**SEM string**

**SEM Main structure**

**SEM Private/Shared table**

**SM Disk Frame Heap**

**SM SFN array**

**SM Swap Frame**

**TK Exit List record**

**TK dispatch (KM) registers**

**TK LibInit Free Notification record**

**TK LibInit record**

**TK PTDA**

**TK TCB**

**TK TSD**

**VDD block header**

**VDD memory block**

**VDD config.sys string**

**VDD creation tmp allocation**

**VDD Entry Point**

**VDD heap header**

**Heap objects to load VDDs**

**VDD hook**

**VDD Linear Arena header**

**VDD Linear arena Record**

**VDD module record**

**VDD record**

**VDD PDD Entry Point**

**VDD procedure record**

**VDD string**

**VDH fault hook**
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdhalloc</td>
<td>0xffde</td>
<td>VDH services resident memory</td>
</tr>
<tr>
<td>vdhswap</td>
<td>0xffdf</td>
<td>VDH services swappable memory</td>
</tr>
<tr>
<td>vdmalias</td>
<td>0xffe0</td>
<td>VDM Alias</td>
</tr>
<tr>
<td>vmah</td>
<td>0xffe1</td>
<td>VM arena header</td>
</tr>
<tr>
<td>vmal</td>
<td>0xffe2</td>
<td>VM Alias Record</td>
</tr>
<tr>
<td>vmar</td>
<td>0xffe3</td>
<td>VM Arena Record</td>
</tr>
<tr>
<td>vmbmap</td>
<td>0xffe4</td>
<td>VM Location Bitmap</td>
</tr>
<tr>
<td>vmco</td>
<td>0xffe5</td>
<td>VM Context Record</td>
</tr>
<tr>
<td>vmdead</td>
<td>0xffe6</td>
<td>VM Dead Object</td>
</tr>
<tr>
<td>vmhsh</td>
<td>0xffe7</td>
<td>VM Location Hash Table</td>
</tr>
<tr>
<td>vmkrhb</td>
<td>0xffe8</td>
<td>VM &quot;UNKNOWN&quot; busy KRHB</td>
</tr>
<tr>
<td>vmkrhf</td>
<td>0xffe9</td>
<td>VM free KRHB</td>
</tr>
<tr>
<td>vmkrhl</td>
<td>0xffea</td>
<td>VM end KRHB</td>
</tr>
<tr>
<td>vmkrhro</td>
<td>0xffeb</td>
<td>VM Public Kernel Resident R/O Heap</td>
</tr>
<tr>
<td>vmkrhrw</td>
<td>0xffec</td>
<td>VM Public Kernel Resident R/W Heap</td>
</tr>
<tr>
<td>vmkshd</td>
<td>0xffed</td>
<td>VM Swappable Heap Descriptor</td>
</tr>
<tr>
<td>vmkshro</td>
<td>0xffee</td>
<td>VM Public Kernel Swappable R/O Heap</td>
</tr>
<tr>
<td>vmkshrw</td>
<td>0xffef</td>
<td>VM Public Kernel Swappable R/W Heap</td>
</tr>
<tr>
<td>vmllock</td>
<td>0xfff0</td>
<td>VM long term lock manager</td>
</tr>
<tr>
<td>vmob</td>
<td>0xfff1</td>
<td>VM Object Record</td>
</tr>
<tr>
<td>vmsgs</td>
<td>0xfff2</td>
<td>VM Screen Group Switch record</td>
</tr>
<tr>
<td>vmbmpl6</td>
<td>0xfff3</td>
<td>VM Temp buf (BMP16)</td>
</tr>
<tr>
<td>shrind</td>
<td>0xfff4</td>
<td>reserved for shared indicator</td>
</tr>
<tr>
<td>give</td>
<td>0xfff5</td>
<td>giveable segment</td>
</tr>
<tr>
<td>get</td>
<td>0xfff6</td>
<td>gettable segment</td>
</tr>
<tr>
<td>giveget</td>
<td>0xfff7</td>
<td>giveable and gettable segment</td>
</tr>
<tr>
<td>preload</td>
<td>0xfff8</td>
<td>Loader's preload object</td>
</tr>
</tbody>
</table>
DevHlp_SchedClock 0x0  Called each timer tick
DevHlp_DevDone 0x1  Device I/O complete
DevHlp_Yield 0x2  yield CPU if resched set
DevHlp_TCYield 0x3  yield to time critical task
DevHlp_ProcBlock 0x4  Block on event
DevHlp_ProcRun 0x5  Unblock process
DevHlp_SemRequest 0x6  claim a semaphore
DevHlp_SemClear 0x7  release a semaphore
DevHlp_SemHandle 0x8  obtain a semaphore handle
DevHlp_PushRequest 0x9  Push the request
DevHlp_PullRequest 0xA  Pull next request from Q
DevHlp_PullParticular 0xB  Pull a specific request
DevHlp_SortRequest 0xC  Push request in sorted order
DevHlp_AllocReqPacket 0xD  allocate request packet
DevHlp_FreeReqPacket 0xE  free request packet
DevHlp_QueueInit 0xF  Init/Clear char queue
DevHlp_QueueFlush 0x10  flush queue
DevHlp_QueueWrite 0x11  Put a char in the queue
DevHlp_QueueRead 0x12  Get a char from the queue
DevHlp_Lock 0x13  Lock segment
DevHlp_Unlock 0x14  Unlock segment
DevHlp_PhysToVirt 0x15  convert physical address to virtual
DevHlp_VirtToPhys 0x16  convert virtual address to physical
DevHlp_PhysToUVirt 0x17  convert physical to LDT
DevHlp_AllocPhys 0x18  allocate physical memory
DevHlp_FreePhys 0x19  free physical memory
DevHlp_SetROMVector 0x1A  set a ROM service routine vector
DevHlp_SetIRQ 0x1B  set an IRQ interrupt
DevHlp_UnSetIRQ 0x1C  unset an IRQ interrupt
DevHlp_ResetTimer 0x1D  reset timer request handler
DevHlp_ResetTimer 0x1E  unset timer request handler
DevHlp_MonitorCreate 0x1F  create a monitor
DevHlp_Register 0x20  install a monitor
DevHlp_DeRegister 0x21  remove a monitor
DevHlp_MonWrite 0x22  pass data records to monitor
DevHlp_MonFlush 0x23  remove all data from stream
DevHlp_GetDOSVar 0x24  Return pointer to DOS variable
DevHlp_SendEvent 0x25  an event occurred
DevHlp_ROMCritSection 0x26  ROM Critical Section
DevHlp_VerifyAccess 0x27  Verify access to memory
DevHlp_RAS 0x28  Put info in RAS trace buffer
DevHlp_ABIOSGetParms 0x29  Get ABIOS CallingParms
DevHlp_AttachDD 0x2A  Attach to a device driver
DevHlp_InternalError 0x2B  Signal an internal error
DevHlp_ModifyPriority 0x2C  Undocumented (used by PM)
DevHlp_AllocGDTSelector 0x2D  Allocate GDT Selectors
DevHlp_PhysToGDTSelector 0x2E  Convert phys addr to GDT sel
DevHlp_RealToProt 0x2F  Change from real to protected mode
DevHlp.ProtToReal 0x30  Change from protected to real mode
DevHlp_EOI 0x31  Send EOI to PIC
DevHlp_UnPhysToVirt 0x32  mark completion of PhysToVirt
DevHlp_TickCount 0x33  modify timer
DevHlp_GetLIDEntry 0x34  Obtain Logical ID
DevHlp_FreeLIDEntry 0x35  Release Logical ID
DevHlp_ABIOSCall 0x36  Call ABIOS
DevHlp_ABIOSCommonEntry 0x37  Invoke Common Entry Point
DevHlp_GetDeviceBlock 0x38  Get ABIOS Device Block
DevHlp_RegisterStackUsag 0x3A  Register for stack usage
DevHlp_LogEntry 0x3B  Place data in log buffer
DevHlp_VideoPause 0x3C  Video pause on/off
DevHlp_Save_Message 0x3D  Save msg in SysInit Message Table
DevHlp_SegRealloc 0x3E  Realloc DD protect mode segment
DevHlp_PutWaitingQueue 0x3F  Put I/O request on waiting queue
DevHlp_GetWaitingQueue 0x40  Get I/O request from waiting queue
DevHlp_PhysToSys 0x41  Address conversion for the AOX
DevHlp_PhysToSysHook 0x42  Address conversion for the AOX
DevHlp_RegisterDeviceClass 0x43  Register DC entry point
DevHlp_RegisterPDD 0x50  Register PDD entry point with VDM manager for later PDD-VDD communication
DevHlp_RegisterBeep 0x51  register PTD beep service entry point with kernel
DevHlp_Beep 0x52  preempt beep service via PTD
DevHlp_FreeGDTSelector 0x53  Free allocated GDT selector
DevHlp_PhysToGDTSel 0x54  Convert Phys Addr to GDT sel with given access
DevHlp_VMLock 0x55  Lock linear address range
DevHlp_VMUnlock 0x56  Unlock address range
DevHlp_VMAlloc 0x56  Allocate memory
DevHlp_VMFree 0x58  Free memory or mapping
DevHlp_VMProcessToGlobal 0x59  Create global mapping to process memory
DevHlp_VMGlobalToProcess 0x5A  Create process mapping to global memory
DevHlp_VirtToLin 0x5B Convert virtual address to linear
DevHlp_LinToGDTSelector 0x5C Convert linear address to virtual
DevHlp_GetDescInfo 0x5D Return descriptor information
DevHlp_LinToPageList 0x5E build pagelist array from lin addr
DevHlp_PageListToLin 0x5F map page list array to lin addr
DevHlp_PageListToGDTSelector 0x60 map page list array to GDT sel.
DevHlp_RegisterTmrDD 0x61 Register TMR Device Driver.
DevHlp_RegisterPerfCtrs 0x62 Register device driver perf. ctrs (PVW).
DevHlp_AllocateCtxHook 0x63 Allocate a context hook
DevHlp_FreeCtxHook 0x64 Free a context hook
DevHlp_ArmCtxHook 0x65 Arm a context hook
DevHlp_VMSetMem 0x66 commit/decommit memory
DevHlp_OpenEventSem 0x67 open an event semaphore
DevHlp_CloseEventSem 0x68 close an event semaphore
DevHlp_PostEventSem 0x69 post an event semaphore
DevHlp_ResetEventSem 0x6A reset an event semaphore
DevHlp_RegisterFreq 0x6B register PTD freq service entry point with kernel
DevHlp_DynamicAPI 0x6C add a dynamic API
DevHlp_ProcRun2 0x6D Unblock process via procrun2
DevHlp_CreateInt13VDM 0x6E Create Int13 VDM (Internal Only)
DevHlp_RegisterKrn1Exit 0x6F Used to capture Kernel Exits
DevHlp_PMPostEventSem 0x70 PM Post Event Semaphore
DevHlp_AcquireSpinLock 0x71 acquire Spin Lock (SMP only)
DevHlp_ReleaseSpinLock 0x72 release Spin Lock (SMP only)
DevHlp_InitIntMouseCursorData 0x73 Initialize Mouse/Cursor Data (SMP only)
DevHlp_StartIntMouseCursor 0x74 Start Int Time Mouse/Cursor (SMP only)
DevHlp_EndIntMouseCursor 0x75 End Int Time Mouse/Cursor (SMP only)
DevHlp_Port:\(O\) 0x76 Port I/O (SMP only)
DevHlp_GetIRQMask 0x77 Set/Unset an IRQ Mask (SMP only)
DevHlp_GetIRQMask 0x78 Retrieve an IRQ Mask state (SMP only)
DevHlp_CreateSpinLock 0x79 create Spin Lock (SMP only)
DevHlp_FreeSpinLock 0x7A free Spin Lock (SMP only)
DevHlp_KillProc 0x7D Kill Proc
DevHlp_QSysState 0x7E Query System State
DevHlp_OpenFile 0x7F Ring-0 File system Write
DevHlp_CloseFile 0x80 Ring-0 File system Seek
DevHlp_ReadFile 0x81 Ring-0 File system Read
DevHlp_ReadFileAt 0x82 File system Read at (seek)
Device Driver Strategy Commands Cross-reference

The following table is a cross-reference for Device Driver strategy routine command codes. The command code is located in the PktCmd field of the Strategy 1 Request Packet.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDInit</td>
<td>0x0</td>
<td>INIT command</td>
</tr>
<tr>
<td>CMDMedChk</td>
<td>0x1</td>
<td>Media Check</td>
</tr>
<tr>
<td>CMDBldBPB</td>
<td>0x2</td>
<td>build BPB</td>
</tr>
<tr>
<td>CMDIOCTLR</td>
<td>0x3</td>
<td>reserved for 3.x compatibility</td>
</tr>
<tr>
<td>CMDINPUT</td>
<td>0x4</td>
<td>read data from device</td>
</tr>
<tr>
<td>CMDNDR</td>
<td>0x5</td>
<td>non-destructive read</td>
</tr>
<tr>
<td>CMDInputS</td>
<td>0x6</td>
<td>input status</td>
</tr>
<tr>
<td>CMDInputF</td>
<td>0x7</td>
<td>input flush</td>
</tr>
<tr>
<td>CMDOUTPUT</td>
<td>0x8</td>
<td>write data to device</td>
</tr>
<tr>
<td>CMDOUTPUTV</td>
<td>0x9</td>
<td>write data and verify</td>
</tr>
<tr>
<td>CMDOutputS</td>
<td>0xa</td>
<td>output status</td>
</tr>
<tr>
<td>CMDOutputF</td>
<td>0xb</td>
<td>output flush</td>
</tr>
<tr>
<td>CMDIOCTLW</td>
<td>0xc</td>
<td>reserved for 3.x compatibility</td>
</tr>
<tr>
<td>CMDOpen</td>
<td>0xd</td>
<td>device open</td>
</tr>
<tr>
<td>CMDClose</td>
<td>0xe</td>
<td>device close</td>
</tr>
<tr>
<td>CMDRemMed</td>
<td>0xf</td>
<td>is media removable</td>
</tr>
<tr>
<td>CMDGenIOCTL</td>
<td>0x10</td>
<td>Generic IOCTL</td>
</tr>
<tr>
<td>CMDResetMed</td>
<td>0x11</td>
<td>reset media uncertain</td>
</tr>
<tr>
<td>CMDGetLogMap</td>
<td>0x12</td>
<td></td>
</tr>
<tr>
<td>CMDSetLogMap</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td>CMDDeInstall</td>
<td>0x14</td>
<td>De-Install driver</td>
</tr>
<tr>
<td></td>
<td>0x15</td>
<td>reserved</td>
</tr>
<tr>
<td>CMDPartfixeddisks</td>
<td>0x16</td>
<td>Partitionable Fixed Disks</td>
</tr>
<tr>
<td>CMDGetfd_logunitsmap</td>
<td>0x17</td>
<td>Get Fixed Disk/Logical Unit Map</td>
</tr>
<tr>
<td>CMDInputBypass</td>
<td>0x18</td>
<td>cache bypass read data</td>
</tr>
<tr>
<td>CMDOutputBypass</td>
<td>0x19</td>
<td>cache bypass write data</td>
</tr>
<tr>
<td>CMDOutputBypassV</td>
<td>0x1a</td>
<td>cache bypass write data and verify</td>
</tr>
<tr>
<td>CMDInitBase</td>
<td>0x1b</td>
<td>INIT command for base DDs</td>
</tr>
<tr>
<td>CMDShutdown</td>
<td>0x1c</td>
<td></td>
</tr>
<tr>
<td>CMDGetDevSupport</td>
<td>0x1d</td>
<td>query for extended capability</td>
</tr>
</tbody>
</table>
### System Ordinal Cross-reference

The following table is a cross-reference for System Entry points by Ordinal number.

<table>
<thead>
<tr>
<th>Ord</th>
<th>Entry Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOSCREATETHREAD</td>
</tr>
<tr>
<td>2</td>
<td>DOSCWAIT</td>
</tr>
<tr>
<td>3</td>
<td>DOSENTERCRTSEC</td>
</tr>
<tr>
<td>4</td>
<td>DOSEXECPCGM</td>
</tr>
<tr>
<td>5</td>
<td>DOSEXIT</td>
</tr>
<tr>
<td>6</td>
<td>DOSEXITCRITICALSEC</td>
</tr>
<tr>
<td>7</td>
<td>DOSEXITLIST</td>
</tr>
<tr>
<td>8</td>
<td>DOSGETINFOSEG</td>
</tr>
<tr>
<td>9</td>
<td>DOSGETPRITY</td>
</tr>
<tr>
<td>10</td>
<td>DOSKILLPROCESS</td>
</tr>
<tr>
<td>11</td>
<td>DOSSETPRITY</td>
</tr>
<tr>
<td>12</td>
<td>DOSTRACE</td>
</tr>
<tr>
<td>13</td>
<td>DOSHOLDSIGNAL</td>
</tr>
<tr>
<td>14</td>
<td>DOSSETSIGNHANDLER</td>
</tr>
<tr>
<td>15</td>
<td>DOSFLAGPROCESS</td>
</tr>
<tr>
<td>16</td>
<td>DOSMAKEPIPE</td>
</tr>
<tr>
<td>17</td>
<td>DOSISYSSEMCLEAR</td>
</tr>
<tr>
<td>18</td>
<td>DOSISEMREQUEST</td>
</tr>
<tr>
<td>19</td>
<td>DOSISYSEMSET</td>
</tr>
<tr>
<td>20</td>
<td>DOSISEMSETWAIT</td>
</tr>
<tr>
<td>21</td>
<td>DOSSEMSETWAIT</td>
</tr>
<tr>
<td>22</td>
<td>DOSMUXSEMWAIT</td>
</tr>
<tr>
<td>23</td>
<td>DOSCLOSESEM</td>
</tr>
</tbody>
</table>
24   DOSCREATESEM
25   DOSOPENSEM
26   DOSRESUMETHREAD
27   DOSUSPENDTHREAD
28   DOSSETDATETIME
29   DOSTIMERASYNC
30   DOSTIMERSTART
31   DOSTIMERSTOP
32   DOSSLEEP
33   DOSGETDATETIME
34   DOSALLOCSEG
35   DOSALLOCSSHRSSEG
36   DOSGETSSHRSSEG
37   DOSGIVESEG
38   DOSREALLOCSEG
39   DOSFREESSEG
40   DOSALLOCCHUGE
41   DOSGETHUGESHIFT
42   DOSREALLOCCHUGE
43   DOSCREATECSALIAS
44   DOSLOADMODULE
45   DOSGETPROCADDR
46   DOSFREEMODULE
47   DOSGETMODHANDLE
48   DOSGETMODNAME
49   DOSGETMACHINEMODE
50   DOSBEEP
51   DOSCLIACCESS
52   DOSDEVCONFIG
53   DOSDEVIOTL
54   DOSSGSWITCH
55   DOSSGSWGITCHME
56   DOSBUFFRESET
57   DOSCHDIR
58   DOSCHGFILEPTR
59   DOSCLOSE
60   DOSDELETE
61   DOSDUPHANDLE
62   DOSFILELOCKS
63   DOSFINDCLOSE
DOSFINDFIRST
DOSFINDDNEXT
DOSMKDIR
DOSMOVE
DOSNEWSIZE
DOSPORTACCESS
DOSOPEN
DOSCURDIR
DOSCURDISK
DOSQHANDSTATE
DOSQFILEINFO
DOSQFILEMODE
DOSQFSINFO
DOSHANDTYPE
DOSQVERIFY
DOSREAD
DOSRMDIR
DOSSELECTDISK
DOSSETFHANDSTATE
DOSSETFILEINFO
DOSSETFILEMODE
DOSSETMAXFH
DOSSETVERIFY
DOSWRITE
DOSSYSTEMSERVICE
DOSSETVEC
DOSSTRACE
DOSGETENV
DOSGETVERSION
DOSQTRACEINFO
DOSGETPID
DOSOPEN2
DOSLIBINIT
DOSGETFSINFO
DOSQPATHINFO
DOSDEVIOCTL2
DOSICANONICALIZE
DOSSETFGND
DOSWAPTASKINIT
103  DOSREADPHYS
104  DOSSETPATHINFO
105  DSSGSWITCHPROC2
106  STRUCHECK
107  STRURESUPDATE
108  DOSISETRELMAXFH
109  DOSIDEVIOCTL
110  DOS32FORCEDDELETE
111  DOS32KILLTHREAD
112  DOSQUERYRASINFO
113  DOS32DUMPPROCESS
114  DOS32SUPPRESSPOPUPS
118  DOSOPEN2COMPT
119  DOSGETSTDIA
120  DOSERROR
121  DOSGETSEG
122  DOSLOCKSEG
123  DOSUNLOCKSEG
124  DSSGSWITCHPROC
125  DOSIRAMSEM_WAKE
126  DSSIZESEG
127  DOSMEMAVAIL
128  DOSIRAMSEMREQUEST
129  DOSPHYSICALDISK
130  DOSGETCP
131  DOSISETCP
132  DOSGLOBALSEG
133  DOSPROFILE
134  DOSSEND_SIGNAL
135  DOSHUGESHIFT
136  DOSHUGEINC
137  DOSREAD
138  DOSWRITE
139  DOSERRCLASS
140  DOSSEMREQUEST
141  DOSSEMCLEAR
142  DOSSEMWAIT
143  DOSSEMSET
144  DOSEXEC_PGM
145  DOSCREATETHREAD
DOSSUBSET
DOSSUBALLOC
DOSSUBFREE
DOSREADASYNC
DOSWRITEASYNC
DOSSEARCHPATH
DOSSCANENV
DOSSETTCP
DOSQPROCSTATUS
DOSGETRESOURCE
DOSGETPPID
DOSCALLOBACK
DOSICALLOBACK
DOSRETFORWARD
DOSR2STACKREALLOC
DOSFSRAMSEMCLEAR
DOSQAPPTYPE
DOSSETPROCCP
DOSDYNAMICTRACE
DOSQSYSINFO
DOSIMAKENMPIPE
DOSICALLNMPIPE
DOSCONNECTNMPIPE
DOSIDISCONNECTNMPIPE
DOSIPEEKNMPIPE
DOSIQNMPIPEINFO
DOSIQNMPHANDBSTATE
DOSISETNMPHANDBSTATE
DOSITRANSACTIONNMPIPE
DOSIWAITNMPIPE
DOSISETNMPIPESEM
DOSIQNMPPIPESEMSTATE
DOSIRAWREADNMPIPE
DOSIRAWWRITENMPIPE
DOSFSATTACH
DOSQFSATTACH
DOSFSCTL
DOSFINDFIRST2
185 DOSMKDIR2
186 DOSFILEIO
187 DOSFINDNOTIFYCLOSE
188 DOSFINDNOTIFYFIRST
189 DOSFINDNOTIFYNEXT
190 DOSSETTRACEINFO
191 DOSEDITNAME
192 DOSLOGMODE
193 DOSLOGENTRY
194 DOGETLOGBUFFER
195 DOSLOGREGISTER
196 DOSLOGREAD
197 DOSFINDFROMNAME
198 DOSOPLOCKRELEASE
199 DOSOPLOCKWAIT
200 DOSCOPY
201 DOSCOPY2
202 DOSIQAPPTYPE
203 DOSFORCEDDELETE
204 DOSENUMATTRIBUTE
205 DOSOPLOCKSHUTDOWN
206 DOSSHUTDOWN
207 DOSGETRESOURCE2
208 DOSFREERESOURCE
209 DOS32SETMAXFH
210 DOS32SETVERIFY
211 DOS32ERRCLASS
212 DOS32ERROR
213 DOSCREATEVDM
214 DOSMAXPATHLEN
215 DOSPAGESIZE
216 DOSLOCALINFO
217 DOSGLOBALINFO
218 DOS32SETFILEINFO
219 DOS32SETPATHINFO
220 DOS32SETDEFAULTDISK
221 DOS32SETFHSTATE
222 DOS32SETFSINFO
223 DOS32QUERYPATHINFO
224 DOS32QUERYHTYPE
DOS32QUERYVERIFY
DOS32DELETEDIR
DOS32SCANENV
DOS32SEARCHPATH
DOS32SLEEP
DOS32GETDATETIME
DOS32DEVCONFIG
DOS32ENTERCRITSEC
DOS32EXICTCRITSEC
DOS32EXIT
DOS32KILLPROCESS
DOS32SETPRIORITY
DOS32RESUMETHREAD
DOS32SUSPENDTHREAD
DOS32CREATEPIPE
DOS32CALLNPIPE
DOS32CONNECTNPIPE
DOS32DISCONNECTNPIPE
DOS32CREATENPIPE
DOS32PEEKNPIPE
DOS32QUERYNPIPESTATE
DOS32RAWREADNPIPE
DOS32RAWWRITENPIPE
DOS32QUERYNPIPEINFO
DOS32QUERYNPIPESEMEMSTATE
DOS32SETNPIPESTATE
DOS32SETNPIPESESEM
DOS32TRANSACTNPIPE
DOS32WAITNPIPE
DOS32RESETBUFFER
DOS32SETCURRENTDIR
DOS32SETFILEPTR
DOS32CLOSE
DOS32COPY
DOS32DELETE
DOS32DUPHANDLE
DOS32EDITNAME
DOS32FINDCLOSE
DOS32FINDFIRST
DOS32FINDNEXT
DOSOPENVDD
DOSREQUESTVDD
DOSCLOSEVDD
DOS32FSATTACH
DOS32CREATEDIR
DOS32MOVE
DOS32SETFILESIZE
DOS32OPEN
DOS32QUERYCURRENTDIR
DOS32QUERYCURRENTDISK
DOS32QUERYFHSTATE
DOS32QUERYFSATTACH
DOS32QUERYFSINFO
DOS32QUERYFILEINFO
DOS32WAITCHILD
DOS32READ
DOS32WRITE
DOS32EXECPGM
DOS32DEVIIOCTL
DOS32FSCTL
DOS32BEEP
DOS32PHYSICALDISK
DOS32SETCP
DOS32SETPROCESSCP
DOS32STOPTIMER
DOS32QUERYCP
DOS32SETDATETIME
THK32ALLOCBLOCK
THK32FREEBLOCK
THK32R3DS
DOS32EXITLIST
DOS32ALLOCPROTECTEDMEM
DOS32ALIASMEM
DOS32ALLOCMEM
DOS32ALLOCSHAREDMEM
DOS32GETNAMEDSHAREDMEM
DOS32GETSHAREDMEM
DOS32GIVESHAREDMEM
DOS32GIVESHAREDMEM
DOS32FREEMEM
305  DOS32SETMEM
306  DOS32QUERYMEM
307  DOS32QUERYMEMSTATE
308  DOS32OPENVDD
309  DOS32REQUESTVDD
310  DOS32CLOSEVDD
311  DOS32CREATETHREAD
312  DOS32GETINFOBLOCKS
313  DOSALLOCPROTSEG
314  DOSALLOCSHRPROTSEG
315  DOSALLOCPROTHUGE
316  DOS32DYNAMICTRACE
317  DOS32DEBUG
318  DOS32LOADMODULE
319  DOS32QUERYMODULEHANDLE
320  DOS32QUERYMODULENAME
321  DOS32QUERYPROCADDR
322  DOS32FREEMODULE
323  DOS32QUERYAPPTYPE
324  DOS32CREATEEVENTSEM
325  DOS32OPENEVENTSEM
326  DOS32CLOSEEVENTSEM
327  DOS32RESETEVENTSEM
328  DOS32POSTEVENTSEM
329  DOS32WAITEVENTSEM
330  DOS32QUERYEVENTSEM
331  DOS32CREatemutexSEM
332  DOS32OPENMUTEXSEM
333  DOS32CLOSEMUTEXSEM
334  DOS32REQUESTMUTEXSEM
335  DOS32RELEASEMUTEXSEM
336  DOS32QUERYMUTEXSEM
337  DOS32CREatemuxwaitSEM
338  DOS32OPENMUXWAITSEM
339  DOS32CLOSEMUXWAITSEM
340  DOS32WAITMUXWAITSEM
341  DOS32ADDMUXWAITSEM
342  DOS32DELETEMUXWAITSEM
343  DOS32QUERYMUXWAITSEM
DOS32SUBSETMEM
DOS32SUBALLOCMEM
DOS32SUBFREEMEM
DOS32SUBUNSETMEM
DOS32QUERYSYSINFO
DOS32WAITTHREAD
DOS32ASYNCTIMER
DOS32STARTTIMER
DOS32GETRESOURCE
DOS32FREERESOURCE
DOS32SETEXCEPTIONHANDLER
DOS32UNSETEXCEPTIONHANDLER
DOS32RAISEEXCEPTION
DOS32UNUNINDEXCEPTION
DOS32QUERYPAGEUSAGE
DOSQUERYMODFROMCS
DOS32QUERYMODFROMEIP
DOSFPDATAAREA
DOS32TMRQUERYFREQ
DOS32TMRQUERYTIME
DOS32ALIASPERFCTRS
DOS32CONFIGUREPERF
DOS32DECONPERF
DOS32REGISTERPERFCTRS
DOS32QUERYSYSSTATE
DOS32FLATCS
DOS32FLATDS
DOS32QUERYABIOSUPPORT
DOS32ENUMATTRIBUTE
DOS32QUERYDOSPROPERTY
DOS32SETDOSPROPERTY
DOSQUERYDOSPROPERTY
DOSSETDOSPROPERTY
DOS32PROFILE
DOS32SETSIGNALEXCEPTIONFOC
DOS32SENDSIGNALEXCEPTION
DOS32ENTERMUSTCOMPLETE
DOS32EXITMUSTCOMPLETE
DOS32SETRELMAXFH
MSGPUTMESSAGE
MSGTRUEGETMESSAGE
MSGINSMESSAGE
MSG32INSERTMESSAGE
MSG32PUTMESSAGE
MSG32TRUEGETMESSAGE
MSGQUERYMESSAGECP
MSG32QUERYMESSAGECP
NLSCASEMAP
NLGETCOLLATE
NLGETCTRINFO
NLGETDBCSEV
NL32QUERYCTRYINFO
NL32QUERYDBCSENV
NL32MAPCASE
NL32QUERYCOLLATE
NPIPEMAKENMPIPE
NPIPEQNMPIPEINFO
NPIPECONNECTNMPIPE
NPIPEDISCONNECTNMPIPE
NPIPEQNMPHANDSTATE
NPIPESETNMPHNDSTATE
NPIPEPEEKNMPIPE
NPIPEWAITNMPIPE
NPIPETRANSACTNMPIPE
NPIPECALLNMPIPE
NPIPERAWREADNMPIPE
NPIPERAWRITENMPIPE
NPIPESETNMPPIPESEM
NPIPEQNMPPIPESEMSSTATE
HFSSTARTL2ASYWRITER
QUEINSTANCEDATA
DOS32SHUTDOWN
DOS32ICACHEMODULE
DOS32REPLACEMODULE
DOS32ACKNOWLEDGESIGNALEXEC
DOS32TIB
DOSTMRQUERYFREQ
DOSTMRQUERYTIME
DOSREGISTERPERFCTRS
423  DOSFLATTOSEL
424  DosSeltOFlat
425  DOS32FLATTOSEL
426  DOS32SeltOFlat
427  DOSIODELAYCNT
428  DOS32SETFILELOCKS
429  DOS32CANCELLOCKREQUEST
430  LOGOPEN
431  LOGCLOSE
432  LOGADDENTRIES
433  LOGGETENTRIES
434  LOGSETSTATE
435  LOGSETNAME
436  LOGQUERYSTATE
437  DOSOPENCHANGENOTIFY
438  DOSRESETCHANGENOTIFY
439  DOSCLOSECHANGENOTIFY
440  DOS32OPENCHANGENOTIFY
441  DOS32RESETCHANGENOTIFY
442  DOS32CLOSECHANGENOTIFY
443  DOSQUERYABIOSSUPPORT
444  DOS32FORCESYSTEMDUMP
454  DOS32ALLOCTHREADLOCALMEMORY
455  DOS32FREETHREADLOCALMEMORY
460  DOS32VERIFYPDIPTID
464  PTDA_LANMAN_SEC
465  PTDA_PID
466  SAS_SEL
467  TCB_OPCOOKIE
468  TCB_OPFLAGS
469  TCB_NEWFLAGS
470  TCB_USER_ID
471  TCB_PROC_ID
472  TCB_FSHARING
473  TCB_SRVATTRIB
474  TCB_ALLOWED
475  TCB_PRTC
476  TCB_NUMBER
477  TCB_THISSFT
478  TCB_THISCDS
TKOPTDA
PTDA CRITSEC
PTDA HOLDSIGCNT
PTDA PPTDAPARENT
PTDA PGDATA
PTDA HANDLE
PTDA MODULE
PTDA LDTHANDLE
PTDA CODEPAGE_TAG
PTDA JFN_LENGTH
PTDA JFN PTABLE
PTDA JFN FLG_PTR
PTDA EXTERN LOCUS
PTDA EXTERN
PTDA EXTERN_ACTION
PTDA EXTERN_CLASS
PTDA PPID
PTDA PROCTYPE
PTDA CURRTCB
PTDA CURRTSD
PTDA SIGNATURE
DOS32EXCEPTIONCALLBACK
DOS32R3EXCEPTIONDISPATCHER
DOSLIBIDISP
DOSLIBIDISP16
DOSLIBIDISP32
DOSR3EXITADDR
DOS32R3EXITADDR
DOS32IREAD
DOS32IWRITE
DOSISETFILEINFO
DOSISETPATHINFO
DOSIFINDNEXT
DOS32QUERYRESOURCESIZE
DOSQUERYRESOURCESIZE
DOSPMSEMWAIT
DOSPMMUXSEMWAIT
THK16 UNITHUNK
HT16 STARTUP
DOS32INITIALIZEPORTHOLE
DOS32QUERYHEADERINFO
DOSINITIALIZEPORTHOLE
DOSQUERYHEADERINFO
MON32MONREAD
DOS32QUERYPROCTYPE
DOSQUERYPROCTYPE
MON32MONWRITE
DOSISIGDISPATCH
DOS32DLLTERMIDISP
DOS32RAISEEXCEPTION
DOS32QUERYFHSTATE
DOS32ISETFHSTATE
DOSLDTSEL
DOS32R3FRESTOR
DOSIFINDFIRST
DOS32IPROTECTWRITE
DOSIPROTECTSETFILEINFO
DOS32IPROTECTSETFILEINFO
DOS32IPROTECTSETFHSTATE
DOS32IPROTECTQUERYFHSTATE
DOS32IPROTECTSETFILEPTR
DOSPROTECTCLOSE
DOSPROTECTFILEIO
DOSPROTECTFILELOCKS
DOSIPROTECTREAD
DOSPROTECTWRITE
DOSPROTECTNEWSIZE
DOSPROTECTOPEN
DOSPROTECTQFHANDSTATE
DOSPROTECTSETFHANDSTATE
DOSPROTECTQFILEINFO
DOSPROTECTSETFILEINFO
DOSPROTECTCHGFILEPTR
DOSPROTECTENUMATTRIBUTE
DOS32PROTECTENUMATTRIBUTE
DOS32PROTECTOPEN
DOS32PROTECTQFILEINFO
DOS32PROTECTSETFILELOCKS
DOS32PROTECTFILESIZE
DOS32PROTECTREAD
DOS32PROTECTWRITE
DOS32PROTECTSETFILEINFO
DOS32PROTECTSETFHSTATE
DOS32PROTECTQUERYFHSTATE
DOS32PROTECTQUERYFILEINFO
DOS32IPROTECTREAD
MSGCLOSEMESSAGEFILE
DOSLDRDIRTYWORKER
DOS16LDRDIRTYWORKER
QUEDOS32READQUEUE
QUEDOS32PURGEQUEUE
QUEDOS32CLOSEQUEUE
QUEDOS32QUERYQUEUE
QUEDOS32PEEKQUEUE
QUEDOS32WRITEQUEUE
QUEDOS32OPENQUEUE
QUEDOS32CREATEQUEUE
SMGDOS32STARTSESSION
SMGDOS32SELECTSESSION
SMGDOS32SETSESSION
SMGDOS32STOPSESSION
SMGREGISTERNOTIFICATION
QUEDOSREADQUEUE
QUEDOSPURGEQUEUE
QUEDOSCLOSEQUEUE
QUEDOSQUERYQUEUE
QUEDOSPEEKQUEUE
QUEDOSWRITEQUEUE
QUEDOSOPENQUEUE
QUEDOSCREATEQUEUE
CHRDOSSMGETME
CHRDOSSMFREEMEM
CHRDOSSMGETSGCB
CHRDOSSMINITSGCB
SMGDOSMSGDOPOPUP
SMGDOSMSWITCH
SMGDOSMSERVEAPPREQ
SMGDOSGETTIMES
769  KBDOOSSETHWID
770  VIODOSENDPOPUP
771  VIODOSGETPHYSBUF
772  VIODOSGETANSI
773  VIODOSFREE
774  VIODOSETANSI
775  VIODOSDEREGISTER
776  VIODOSSCROLLUP
777  VIODOSPRTSC
778  VIODOSGETCURPOS
779  VIODOSWRTCCELLSTR
780  VIODOSPopup
781  VIODOSSCROLLRT
782  VIODOSWRTCHARSTR
783  VIODOSAVS_PRTSC
784  VIODOSETCURPOS
785  VIODOSRFUNBLOCK
786  VIODOSSRFBLOCC
787  VIODOSSCRUNLOCK
788  VIODOSSWRTTY
789  VIODOSSAVE
790  VIODOSGETMODE
791  VIODOSETMODE
792  VIODOSSCRLOCK
793  VIODOSSREADCELLSTR
794  VIODOSSAVREDRAWWAIT
795  VIODOSSWRNATTR
796  VIODOSGETCURTYPE
797  VIODOSSAVREDRAWUNDO
798  VIODOSGETFONT
799  VIODOSSREADCHARSTR
800  VIODOSGETBUF
801  VIODOSETCURTYPE
802  VIODOSETFONT
803  VIODOSHETINIT
804  VIODOSMODEUNDO
805  VIODOSSWSSWITCH
806  VIODOSMODEWAIT
807  VIODOSAVS_PRTSCTOGGLE
808  VIODOSGETCP
809  VIOOSRESTORE
810  VIOOSSETTCP
811  VIOOSSHOWBUF
812  VIOOSSCROLLLF
813  VIOOSREGISTER
814  VIOOSGETCONFIG
815  VIOOSSCROLLDN
816  VIOOSWRKCHARSTRATT
817  VIOOSGETSTATE
818  VIOOSPRTSCTOGGLE
819  VIOOSSETSTATE
820  VIOOSWRNCELL
821  VIOOSWRNCHAR
822  VIOOSHELINIT
823  VIOOSASSOCIATE
824  VIOOSCREATEPS
825  VIOOSDELETESETID
826  VIOOSGETDEVICECELLSIZE
827  VIOOSGETORG
828  VIOOSCREATELOGFONT
829  VIOOSDESTROYSYS
830  VIOOSQUERYSETIDS
831  VIOOSSETSTATE
832  VIOOSQUERYFONTS
833  VIOOSSETDEVICECELLSIZE
834  VIOOSSHOWPS
835  VIOOSGETPSADDRESS
836  VIOOSQUERYCONSOLE
837  VIOOSREDRAWSIZE
838  VIOOSGLOBALREG
839  XVIOSSETCASTATE
840  XVIOSCHECKCHARTYPE
841  XVIOSDESTROYCA
842  XVIOSCREATECA
843  VIOOSCHECKCHARTYPE
844  XVIOSGETCASTATE
845  BVSDOSMAIN
846  BVSDOSREDRAWSIZE
847  BVSDOSGETPTRDRAWNAME
ANSIDOSINJECT
ANSIDOSKEYDEF
ANSIDOSINTERP
BKSOSMAIN
BMSOSMAIN
MOUDOSGETHOTKEY
MOUDOSSETHOTKEY
SMGDOSSMSYSINIT
SMGQHKEYBDHANDLE
SMGQHMOUSEHANDLE
CHRQueueRamSem
CHRArray
CHRPIDArray
CHRInitialized
CHRArraySize
CHRBVSGLOBAL
CHRSMGINSTANCE
CHRBVHINSTANCE
THK32ALLOCMEM
THK32FREEEMEM
THK32ALLOCSTACK
THK32FREESTACK
THK32STRLEN
THK32_UNITHUNK
HT16_CLEANUP
HT32__STARTUP
HT32__CLEANUP
DOS32PMPOSTEVENTSEM
DOS32PMWAITEMVENTSEM
DOS32PMREQUESTMUTEXSEM
DOS32PMWAITMUXWAITSEM
DOS32PM16SEMCHK
THK32ALIASMEM
THK32FREEALIAS
THK32ALLOCVARLEN
THK32HANDLEBOUNDARY
THK32HANDLESTRING
THK32DEALLOC
THK32XHNDLR
DOS32SETTEXTLIBPATH
874  DOS32QUERYEXTLIBPATH
875  DOS32PM16SEMRST
876  DOS32SYSLCTL
998  DOSSETEXTLIBPATH
999  DOSQUERYEXTLIBPATH
1000 T32EXITLIST
1001 T32ALLOCPROTECTDMEM
1002 T32ALIASMEM
1003 T32ALLOCMEM
1004 T32ALLOCSHAREDMEM
1005 T32GETNAMEDIASHAREDMEM
1006 T32GETSHAREDMEM
1007 T32GIVESHAREDMEM
1008 T32FREEMEM
1009 T32SETMEM
1010 T32QUERYMEM
1011 T32QUERYMEMSTATE
1012 T32OPENVDD
1013 T32REQUESTVDD
1014 T32CLOSEVDD
1015 T32CREATETHREAD
1016 T32DYNAMICTRACE
1017 T32DEBUG
1018 T32QUERYPROCADDR
1019 T32CREATEEVENTSEM
1020 T32OPENEVENTSEM
1021 T32CLOSEEVENTSEM
1022 T32RESETEVENTSEM
1023 T32POSTEVENTSEM
1024 T32WAITEVENTSEM
1025 T32QUERYEVENTSEM
1026 T32CREATEMUTEXSEM
1027 T32OPENMUTEXSEM
1028 T32CLOSERMUTEXSEM
1029 T32REQUESTMUTEXSEM
1030 T32RELEASEMUTEXSEM
1031 T32QUERYMUTEXSEM
1032 T32CREATEMUXWAITSEM
1033 T32OPENMUXWAITSEM
1034 T32CLOSEMUXWAITSEM
1035 T32WAITMUXWAITSEM
1036 T32ADDMUXWAITSEM
1037 T32DELETEMUXWAITSEM
1038 T32QUERYMUXWAITSEM
1039 T32QUERYSYSINFO
1040 T32WAITTHREAD
1041 T32GETRESOURCE
1042 T32FREETRESOURCE
1043 T32EXCEPTIONCALLBACK
1044 T32QUERYPAGEUSAGE
1045 T32FORCESYSTEMDUMP
1046 T32ASYNCTIMER
1047 T32STARTTIMER
1048 T32QUERYABIOSUPPORT
1049 T32QUERYMODFROMEIP
1050 T32ALIASPERFCTRS
1051 T32CONFIGUREPERF
1052 T32DECONPERF
1053 T32REGISTERPERFCTRS
1054 T32QUERYSYSSTATE
1055 T32READ
1056 T32WRITE
1057 T32TMRQUERYFREQ
1058 T32TMRQUERYTIME
1059 T32IMONREAD
1060 T32IMONWRITE
1061 T32QUERYRESOURCESIZE
1062 T32PROFILE
1063 T32SETSIGNALEXCEPTIONFOC
1064 T32SENDSIGNALEXCEPTION
1065 T32STARTTIMER
1066 T32STOPTIMER
1067 T32ASYNCTIMER
1068 T32INITIALIZEPORTHOLE
1069 T32QUERYHEADERINFO
1070 T32QUERYPROCYPE
1071 T32EXITMUSTCOMPLETE
1072 T32ICACHEMODULE
1073 T32DLLTERM
OS/2 Fix Pack to Build Level Cross-reference

The following tables cross-references some of the public fix packs and GA versions of OS/2 with their internal kernel build level.

Notes:

Some fix packs use the same kernel build level when updates are confined to modules other than OS2KRNL.

The build level for a system may be determined using the command:

```
VER /R
```

The build level of a system module may be determined using the command:

```
BLDLEVEL  <file name>
```
The fix pack level for a system may be determined using the **SYSLEVEL** command. The following example shows the system level information for OS/2 Warp V4.0 fix pack 6:

```
C:\OS2\INSTALL\SYSLEVEL.FPK
OS/2 Warp 4 Service Level
Version 1.00     Component ID 566933010
Type Fixpak
Current CSD level: XR0M006
Prior   CSD level: XR0M006
```

**OS/2 V2.11 Build Level Cross-reference**

**OS/2 Warp V3.0 Build Level Cross-reference**

**OS/2 Warp V4.0 Build Level Cross-reference**

--------------------------------------------

**OS/2 V2.11 Build Level Cross-reference**

<table>
<thead>
<tr>
<th>Version</th>
<th>Build Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11 GA</td>
<td>6.617</td>
</tr>
<tr>
<td>XR_A076</td>
<td>6.653</td>
</tr>
<tr>
<td>XR_A080</td>
<td>6.653</td>
</tr>
<tr>
<td>XR_A090</td>
<td>6.656</td>
</tr>
<tr>
<td>XR_A092</td>
<td>6.658</td>
</tr>
<tr>
<td>XR_A095</td>
<td>6.661</td>
</tr>
<tr>
<td>XR_A096</td>
<td>6.660</td>
</tr>
<tr>
<td>XR_B097</td>
<td>6.664</td>
</tr>
<tr>
<td>XR_B098</td>
<td>6.665</td>
</tr>
<tr>
<td>XR_B099</td>
<td>6.667</td>
</tr>
<tr>
<td>XR_B100</td>
<td>6.668</td>
</tr>
<tr>
<td>XR_B101</td>
<td>6.669</td>
</tr>
<tr>
<td>XR_B102</td>
<td>6.670</td>
</tr>
<tr>
<td>XR_B103</td>
<td>6.671</td>
</tr>
<tr>
<td>XR_B104</td>
<td>6.672</td>
</tr>
<tr>
<td>XR_B105</td>
<td>6.673</td>
</tr>
<tr>
<td>XR_B106</td>
<td>6.674</td>
</tr>
<tr>
<td>XR_B107</td>
<td>6.675</td>
</tr>
<tr>
<td>XR_B108</td>
<td>6.676</td>
</tr>
<tr>
<td>XR_B109</td>
<td>6.677</td>
</tr>
<tr>
<td>Version</td>
<td>Build Level</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Warp GA</td>
<td>8.162</td>
</tr>
<tr>
<td>Warp Full Pack</td>
<td>8.200</td>
</tr>
<tr>
<td>Warp Connect</td>
<td>8.209</td>
</tr>
<tr>
<td>XR_W005</td>
<td>8.213B</td>
</tr>
<tr>
<td>XR_W007</td>
<td>8.230</td>
</tr>
<tr>
<td>XR_W008</td>
<td>8.230</td>
</tr>
<tr>
<td>XR_W009</td>
<td>8.234</td>
</tr>
<tr>
<td>XR_W010</td>
<td>8.234</td>
</tr>
<tr>
<td>XR_W011</td>
<td>8.235</td>
</tr>
<tr>
<td>XR_W012</td>
<td>8.236</td>
</tr>
<tr>
<td>XR_W013</td>
<td>8.237</td>
</tr>
<tr>
<td>XR_W014</td>
<td>8.238</td>
</tr>
<tr>
<td>XR_W016</td>
<td>8.240</td>
</tr>
<tr>
<td>XR_W017</td>
<td>8.241</td>
</tr>
<tr>
<td>XR_W018</td>
<td>8.242</td>
</tr>
<tr>
<td>XR_W019</td>
<td>8.243</td>
</tr>
<tr>
<td>XR_W020</td>
<td>8.244</td>
</tr>
<tr>
<td>XR_W021</td>
<td>8.245</td>
</tr>
<tr>
<td>XR_W022</td>
<td>8.246</td>
</tr>
<tr>
<td>XR_W023</td>
<td>8.247</td>
</tr>
<tr>
<td>XR_W024</td>
<td>8.248</td>
</tr>
<tr>
<td>XR_W025</td>
<td>8.249</td>
</tr>
<tr>
<td>XR_W026</td>
<td>8.250</td>
</tr>
<tr>
<td>XR_W027</td>
<td>8.251</td>
</tr>
<tr>
<td>XR_W028</td>
<td>8.252</td>
</tr>
<tr>
<td>XR_W029</td>
<td>8.253</td>
</tr>
<tr>
<td>XR_W032U</td>
<td>8.256_uni</td>
</tr>
<tr>
<td>XR_W033U</td>
<td>8.257_uni</td>
</tr>
<tr>
<td>XR_W034U</td>
<td>8.258_uni</td>
</tr>
<tr>
<td>XR_W035U</td>
<td>8.259_uni</td>
</tr>
<tr>
<td>XR_W036U</td>
<td>8.260_uni</td>
</tr>
</tbody>
</table>
# OS/2 Warp V4.0 Build Level Cross-reference

<table>
<thead>
<tr>
<th>Version</th>
<th>Build Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0 GA</td>
<td>9.023</td>
</tr>
<tr>
<td>XR_M000</td>
<td>9.024</td>
</tr>
<tr>
<td>XR_M001</td>
<td>9.025</td>
</tr>
<tr>
<td>XR_M002</td>
<td>9.026</td>
</tr>
<tr>
<td>XR_M003</td>
<td>9.027</td>
</tr>
<tr>
<td>XR_M004</td>
<td>9.028</td>
</tr>
<tr>
<td>XR_M005</td>
<td>9.029</td>
</tr>
<tr>
<td>XR_M006</td>
<td>9.030</td>
</tr>
</tbody>
</table>

---

## Glossary

The following terms and acronyms have been referenced in this document.

- **AAB**
  - Absolute symbol
- **Block Id**
- **BMP**
- **BPB**
- **Breakpoint**
- **cbargs**
- **CBIOS**
- **CDA**
- **CDIB**
- **CDS**
- **CRI**
- **Compatibility Region Mapping Algorithm (CRMA)**
- **Command Subtree (Csid)**
- **Context**
- **DEM**
- **DosHlp**
- **DPB**
- **FAT**
- **FSC**
- **Gate**
- **GDT**
- **Global Info Segment**
- **hal**
- **har**
- **hco**
- **hmte**
- **hob**
- **hptda**
- **HWND**
- **IDT**
- **Interrupt Vector**
- **IPE**
- **JFN**
JFN Table
KSEM
LDT
Local Info Segment
Maps
MFT
MQ
MTE
OTE
PAI
paragraph
PDB
PF
PIB
Pid
Process
Pseudo-Object
PSP
PTDA
PTE
PTree
PWND
QMSG
RIP
RAS
RLR
RMP
SAS
SFN
SFT
SMS
STDA
Scheduler
Session
Screen Group
SGCB
Slot
SMTE
SQMSG
STE
Symbols
Symbol Groups
Task
TCB
Thrashing
Thread
Thunk
TIB
Tid
TLB
TLMA
Tracepoint
TSD
TSS
UVIRT
VMAH
VMLAL
VMAR
VMCO
VMKH
VMOB
VDM
VP
VPB
WND
Zombie

---------------------------------------------

(No title)
The term loader applies to two distinct components under OS/2:

OS2LDR This is the OS2KRNL loader. One of its functions is to load the OS2KRNL module at boot time. After the system has booted OS2LDR provides the CBIOS layer for the kernel.

System Loader This is a component of the kernel. It is responsible for loading program modules, DLLs, Device Drivers and File System Drivers.

The logging facility discussed in this section applies to the System Loader.

(No title)

Throughout this chapter the term debugger is used loosely to mean any of the following where ambiguity is not a problem:

- Debug Kernel (HSTRICT or ALLSTRICT).
- The debugger component within the debug kernel.
- The debugging console.

(No title)

A simple numeric expression is one that resolves to a single integer value, for example:

- $4$
- $55c7$

Compare this with an address expression that has in addition an address operator (&, %, %%, #) and possibly involves more than one integer value, for example:

- &1fc:45
- #1f:445
- %30045
- %%%15c

(No title)

Feature 82818 introduces the Kernel Debugger .MK command. 82818 is supplied as an APAR fix to:

- OS/2 Warp V3.0: as PJ18364 in fix pack 7
- OS/2 V2.11: as PJ16805 in fix pack 90

(No title)

The control block formatting conventions have been chosen to aid the user of the Kernel Debugger and Dump Formatter.
Each control block is presented in tabular form with 5 columns used as follows:

<table>
<thead>
<tr>
<th>name</th>
<th>Field name, usually taken from the C header or MASM include file definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Offset from the beginning of the structure. The offset is of the form ( xy ) where ( x ) is the signed hexadecimal byte offset from the beginning of the structure and ( y ) is the bit offset from the high-order bit of the byte.</td>
</tr>
<tr>
<td>Leng</td>
<td>Hexadecimal length of the field.</td>
</tr>
<tr>
<td>Type</td>
<td>The field type, for the purposes of displaying storage using the D command. The following values are used:</td>
</tr>
<tr>
<td></td>
<td>S  Complex structure. Choose display command to best suit your needs.</td>
</tr>
<tr>
<td></td>
<td>D  Double word. Use DD to format the field correctly.</td>
</tr>
<tr>
<td></td>
<td>W  Word. Use DW to format the field correctly.</td>
</tr>
<tr>
<td></td>
<td>B  Byte. Use DB to format the field correctly.</td>
</tr>
<tr>
<td></td>
<td>A  ASCII byte string. Use DA to format the field correctly.</td>
</tr>
<tr>
<td>blank</td>
<td>A blank value appears when a field does not begin or end on a byte boundary. In this case format the field from the previous field for which a type value is given. Such bit fields are presented in an order assuming this instruction is followed. Attempts to display bit fields in other ways may lead to a great deal of confusion!</td>
</tr>
</tbody>
</table>

Description

Field description taken usually from the header or include file.

A null row is used to indicate an overlay definition of the same control block.

Flag fields are separately formatted in tabular form.

Where a flag field represents a bit mask, the mask is given in hexadecimal and is assumed to indicated that corresponding bits are set to be in effect. Exceptions are specifically noted in the description.

When the flag field take numerical values then they will be shown in either hexadecimal (prefixed with 0x) or decimal depending on the C or MASM definitions.

---------------------------------------------

(No title)

A PM Send Message Structure (SMS) is used by WinSendMessage to enqueue a synchronously sent message. SMSs are chained from the MQ of both the sender and receiver.

See Exploring 32-bit Presentation Manager Under WARP for more information.

---------------------------------------------

(No title)

A PM Message Queue Header (MQ) is used as an anchor for message processing for a given PM Application's message thread. The MQ is created when a threads calls WinCreateMsgQueue.

See Exploring 32-bit Presentation Manager Under WARP for more information.

---------------------------------------------

(No title)
A PM Queue Message (QMSG) is used by WinPostMsg to enqueue an asynchronously sent message to a thread's message queue. QMSGs are chained from the MQ of the receiver in a circular array.

See Exploring 32-bit Presentation Manager Under WARP for more information.

A PM System Queue Message (SQMSG) is used by the PMDD.SYS device driver to enqueue messages, which represent system input activity, to the system input queue.

See Exploring 32-bit Presentation Manager Under WARP for more information.

A pwnd is a 32-bit pointer to a WND structure.

See Exploring 32-bit Presentation Manager Under WARP for more information.

An hwnd is the handle to a WND structure. This is returned to an application when it uses WinCreateWindow and is used for subsequent PM API calls that affect the window.

See Exploring 32-bit Presentation Manager Under WARP for more information.

A PM Application Anchor Block is allocated in the Thread Local Memory Area when a PM application thread creates a message queue. The AAB contains a pointer to the MQ which allows PM to find the MQ in any context. This is particularly useful to the debugger since it also allows the MQ of any PM thread in the system since the TLMA is saved in a thread's TCB.

See Exploring 32-bit Presentation Manager Under WARP for more information.

The Thread Local Memory Area (TLMA) is an area of private arena memory that is instanciated at a thread level. This is achieved by copying the contents of the TLMA to dff:0024 when a thread switch occurs. The TLMA contents are saved in the TCB at TCBTLMA.

Storage is allocated from the TLMA by using the DosAllocThreadLocalMemory API.

See Context Switching for more information.
A PM Window Structure (WND) is used by PM to represent a window. When an application uses WinCreateWindow the WND is created and the hwnd is returned to the user. The WND contains information about a window's hierarchy, and associated MQ.

See Exploring 32-bit Presentation Manager Under WARP for more information.

A tracepoint is designated location in system or application code where the System Trace Facility will gather data for logging by the STDA. Tracepoints may be implemented statically by use of the DosSysTrace API or dynamically through use of the Dynamic Trace Customiser. System defined tracepoints are documented in the System Tracepoints Reference.

The term Zombie is used to describe a terminal condition of a thread or process. There is a strict operating system definition and two colloquial uses:

- The strict system definition refers to a process that has terminated but whose PTDA has been retained on the zombie queue (_pPTDAFirstZombie) because the process status byte (LISEG+0xa) indicates that its parent wishes to collect termination information through DosWaitChild. The dead child is retained on the zombie queue until either the parent dies or issues DosWaitChild.

- Zombie is also commonly used to refer to a terminating thread or process that has blocked after the application has returned to the operating system. Usually this implies a problem freeing memory because one or more pages have been long-term locked by a device driver.

- The third use of zombie refers to any process that is anonymous. Internal thread, VDMs, and terminating threads can be anonymous.

The Compatibility BIOS (CBIOS) is a layer of code in the OS2LDR that presents a hardware independent interface to the BIOS for the OS2KRNL. The interface to the OS2KRNL is provided through a set of entry points called Dos Helper Functions.

A BIOS Parameter Block is used for low level Disk I/O calls to the BIOS. For further information see:

- The .DBPB Command in the Kernel Debugger and Dump Formatter Command Reference.
- The BPB Structure in the System Reference.
Kernel Semaphores are a form of semaphore, similar to the application 32-bit semaphore, used by kernel routines for longer term blocking. Kernel Semaphores provide additional functionality over the simple blocking mechanism, which includes:

- Priority inversion protection.
- Ownership auditability.

For additional information see the following:

- The .D KSEM command.
- The .PB command.
- The KSEM structures in the System Reference.

Fixed Allocation Table (FAT) file system is the default filing system supported by OS/2. Support for FAT is always present, regardless of any installed file systems.

Internal Processing Errors (IPEs) are unrecoverable error conditions detected by the system while running in ring 0. They may arise from inconsistencies detected by the OS/2 Kernel or from traps occurring in any ring 0 code (Kernel, Installable File System Drivers and Device Drivers).

When the system detects an IPE, it enters a routine called panic where an error message is formatted and displayed and the system is halted.

Reliability, Availability and Serviceability (RAS) is an acronym that refers to diagnostic and service support within OS/2. Frequently it is used as a synonym for the adjective diagnostic.

A task is a hardware-architected thread of execution. The INTEL architecture allows for multiple independent tasks to co-exist and provides the task gate mechanism as a means of switching between tasks. Tasks are represented to the hardware by the TSS.

The characteristics of a task are very similar to that of the OS/2 process. Protect-mode processes however, tend to run under a single task in OS/2 and implement switching through the more efficient software-managed context switching mechanism.

Only VDMs and error recovery processes run as independent tasks.
See the INTEL486 Programmer’s Reference for more information.

--------------------------------------------

(No title)

A **gate** descriptor is one that defines to the hardware a means of entering code that executes at a more privileged level of authority. Four types of **gate** are defined:

**Call Gate**

The subject of a **CALL** instruction. Typically used to implement operating system and device driver application programming interfaces (APIs). Device drivers may create **call gates** dynamically using the **DosDynamicAPI** facility.

**Task Gate**

The subject of a call or exception where a (hardware assisted) **task** switch is required.

**Interrupt Gate**

The subject of a hardware or software generated interrupt. Typically an **interrupt gate** will switch execution to an interrupt handler when a device presents an interrupt.

**Trap Gate**

The subject of a trap exception. Used to handle programming errors.

--------------------------------------------

(No title)

A **Virtual Dos Machine (VDM)** is a type of process that runs in an emulated DOS environment using the **DEM** component of OS/2.

--------------------------------------------

(No title)

The **Thread Information Block (TIB)** is a supplemental thread related control block made accessible to ring 3 programs. It contains thread information obtained from the thread’s **TCB** and acts as an anchor for exception-handlers registered for the thread.

The **TIB** may be located from **TCBptib** (**TCB + 0x10**) using the Dump Formatter or Kernel Debugger.

A program gains access to the **TIB** along with the **PIB** by calling the **DosGetInfoBlocks** API.

--------------------------------------------

(No title)

The **Process Information Block (PIB)** is a supplemental process related control block made accessible to ring 3 programs. It contains process status information obtained from the process’ **PTDA**.

The **PIB** may be located from **ptda_avatib** (**PTDA + 0x28**) using the Dump Formatter or Kernel Debugger.

A program gains access to the **PIB** along with the **TIB** by calling the **DosGetInfoBlocks** API.

--------------------------------------------

(No title)

The **Task State Segment (TSS)** is a hardware architected control block that is used for two purposes:
1. To implement the privileged level transition mechanism initiated with a Call Gate instruction.

2. To provide a register save area for hardware task switching initiated with a call to a Task Gate.

In general OS/2 does not use the hardware task switching mechanism, so TSSs are few. It does however use the TSS for implementing Application Programming Interfaces (APIs) in the system.

A TSS may be formatted using the Kernel Debugger and Dump Formatter DT command.

(No title)

The System Trace Data Area (STDA) is a circular buffer used to record trace events. The STDA may be located from the SAS.

(No title)

The Physical Arena Information structure (PAI) describes ranges of physical memory to memory management.

Physical memory ranges mapped by RAM is described by the PAI pointed to by the SAS.

There are two PAIs located at the following symbols:

- **_pgPageablePAI_.** This described ranges of physical memory available for pageable memory allocations. Normally two ranges are described:
  - Below 1Mb
  - 16Mb and above

- **_pgResidentPAI_.** This described ranges of physical memory available for backing resident memory allocations. Normally three ranges are described:
  - Below 1Mb
  - Below 16Mb
  - 16Mb and above

(No title)

The Interrupt descriptor table (IDT) is a hardware architected structure that comprises a table of gate descriptors, one for each interrupt vector. The low numbered entries are defined by the hardware architecture and dedicated to exception management.

Under OS/2 one IDT is allocated for the entire system except for VDMs in which multiple IDTs per VDM are possible. A VDM will allocate one IDT per DPMI client, but if it does not use DPMI then the common system IDT is used. When multiple IDTs are used in a VDM then only the entries not reserved for H/W exceptions and interrupts are allowed to differ.

The Kernel Debugger's V command may be used to intercept system exception handlers.

(No title)
An interrupt vector is presented to the processor when an interrupt is generated either externally by the Programmed Interrupt Controller or internally within the processor chip itself. It is used by the processor as an index into the IDT to determine which interrupt routine should be dispatched.

The processor reserves vectors 0 - 31 to correspond to hardware architected exceptions 0 through 31. Vectors 32 - 255 are reserved for I/O interrupts, which are presented to the processor by the Programmed Interrupt Controller when the one of its IRQ lines is triggered. The correspondence between vectors and IRQs is defined during system initialisation as follows:

- IRQs 0 - 7: vectors 0x50 - 0x57
- IRQs 8 - 15: vectors 0x70 - 0x77

Thus a keyboard interrupt, which is assigned to IRQ 1 under the IBM PC architecture will be handled by the interrupted handler whose interrupt gate is assigned to IDT descriptor 0x51.

See the Dump Formatter and Kernel Debugger DI command for information on displaying IDT entries.

--------------------------------------------

The Local Information Segment is a per-process control block that records the current status of the process. It is imbedded in the PTDA and is also mapped by LDT descriptor 0xdff.

--------------------------------------------

The Global Information Segment (GISEG) is a single instance control block that records the current session status, date and time, trace status and version of the system.

The system maintains two copies of the Global Information Segment to fence against system damage.

The selector for the GISEG may be located from the SAS. See the Dump Formatter and Kernel Debugger .A command.

The GISEG is also mapped locally per-process by LDT descriptor 0xdf4.

--------------------------------------------

A Register Information Packet (RIP) is an entity used to describe the size and offset of a register in a system stack frame. RIPs are located in a CRI.

--------------------------------------------

The Client Register Information (CRI) is a table of Register Information Packets (RIPs) that describe the offset and length of each register that is stored in a ring 0 stack frame on entry to the kernel. This level of indirection allows kernel routines to access entry registers regardless of the stack frame type, of which there are a number, for example:

- System Entry Frames from API calls
- Trap Frames from traps and exceptions
- Interrupt Frames from the interrupt manager
- VDM Stack Frames
Kernel Stack Frames

Each TCB points to a CRI and the associated stack frame from TCB_pcriFrameType (TCB + 0x38) and TCB_pFrameBase (TCB + 0x3c) respectively.

(No title)

Context (or thread context) refers the view of the system any given thread has. Only one thread context may be current at any time.

Switching contexts refers to the process of preparing the system for another thread to run. From an application program's perspective this implies restoring its registers and ring 2 and 3 stacks when it is given the opportunity to run again. From the system's perspective, restoration of an application's registers and stacks is done after the context switch, by the dispatcher, on exiting kernel mode. Not every context switch is followed by exiting kernel mode. For example, if another thread in the same process is in critical section (but blocked) then the new thread enters crt state and the scheduler is called to select yet another thread.

Context switching includes the following system actions:

- Updating GDT descriptor entries 28, 30, 38 and 150b, which point to the current process' LDT,
  the current threads ring 0 stack,
  the current thread's floating point emulator work area,
  the current thread's TIB. (By default the FS selector is loaded with 150b).

  Note: The LDT selector is only updated when the process changes with a context switch, that is, for a process context switch.

- Updating page directory and tables for a process context switch.

- Updating the TR register if the process switch involves a task switch (normally only VDMs).

- Updating the current TSS ring 0 and ring 2 stack addresses.

- Updating system copies of the Global and Local Information Segments.

- Copying the Local Information Segment from the incoming PTDA and the Thread Local Memory Area from the incoming TCB to the segment mapped by LDT selector dfff.

  Note: Besides addressing the current ring 0 stack, selector 30 also addresses the current thread's scheduling control blocks. In particular: the PTDA, TCB and TSD. This is done by aliasing selected address ranges from selector 30 to those of the true PTDA, TCB and TSD in the system arena global memory for the current context. The system defines a dummy module containing a hard-coded PTDA. The symbols of this module have the same name as those of the fields in the PTDA. The system arranges for this to map the PTDA addressed by selector 30. This trick allows the system to refer to PTDA fields for the current context without regard for which process is current, simply by using the field names as public symbols. The user may use the same symbols for referencing the PTDA but these are only valid for the current system context. To access PTDA fields in other contexts the following technique can be used:

```plaintext#
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD   pPTDA   pTCB   Disp  SG  Name
 0025  0004 0002 0004 0001 blk 0300 7b7c8000 7bbc4080 7bbe8a90 1fc4 16 someprog

The current thread slot is 25

We wish to know the thread that has entered critical section in process of thread slot 40. The address of the critical section TCB is saved in ptda_ptcBCritSec and the thread ordinal and slot number are the first two words of the TCB.
```

Note that the current PTDA is located at `PTDA_Start`
DosHlp services comprise a set of hardware dependent service routines established during system initialisation for use by the OS2KRNL and user programs via the OEMHLP$ device driver. Many of the DosHlp services deal with device dependent BIOS behaviour and therefore provide a device independent interface to the BIOS.

A breakpoint is a location in a program where execution is suspended and control is given to a debugging tool. The INTEL architecture supports two implementations of breakpoints for debugging purposes:

The software generated breakpoint using the INT 3 instruction;

The hardware generated breakpoint using the Debugging Registers.

The use of software breakpoints require code modification, whereas the use of debugging registers does not. However, the number of predefined software breakpoints is potentially unlimited whereas there are only 4 breakpoints specifiable using Debugging Registers.

A further distinction between the two types is that software breakpoints only intercept the execution of a particular instruction path, whereas Debugging Registers may be used, in addition, to intercept data fetches and stores from a particular location in virtual memory.

The Kernel Debugger supports both implementations of breakpoints through the use of the:

BR command, which uses Debugging Registers, and

BP command, which uses INT 3 instructions.

The Kernel Debugger limits the predefinition of BP breakpoints to 10, however the programmer may code as may additional INT 3 instructions into his/her program as desired.

The Kernel Debugger refers to breakpoints explicitly set by the BP and BR commands as sticky (implying a certain permanence about them). The G command may have one or more temporary breakpoints established when one or more stop addresses are specified. These are referred to as go breakpoints. Once the Kernel Debugger breaks in go breakpoints are removed. The internal operation of the Kernel Debugger may also necessitate the use of the occasional temporary breakpoints when instruction tracing (see the T and P commands). These are set implicitly and discarded without the user being aware of their existence. Go and temporary breakpoints are created using the INT 3 instruction. Go and sticky BP breakpoints count towards the Kernel Debugger imposed limit of 10, but temporary breakpoints only ever exist singly so do not.

A symbol map is created from symbolic name information generated by a program compiler and converted for used by the Dump Formatter and Kernel Debugger by the linkage editor and MAPSYM utilities. This allows program code and data locations to be referred to by name as well as by address.
A symbol group is the set of symbols that are defined within a program segment. Frequently a program segment is given its own selector at load time.

An symbol is the name given to a program code or data location that has been made public by the programmer. Such symbolic definitions appear in the map file output from the linkage editor. They may be referenced in the Dump Formatter and Kernel Debugger using the L command when the map file is converted to a symbol file using the MAPSYM utility.

An Absolute symbol is a symbolised constant value that has been made public by the programmer. Such symbolic definitions appear in the map file output from the linkage editor and may be referenced in the Dump Formatter and Kernel Debugger using the LA command when the map file is converted to a symbol file using the MAPSYM utility.

A System File Number (SFN) is the system-wide unique handle by which an open file system object is known. It is the offset into the SFT segment that locates the corresponding SFT entry.

Refer to the following for more related information:

.A Kernel Debugger and Dump Formatter command.
.D Kernel Debugger and Dump Formatter command.

A Job File Number (JFN) is a handle for open file system objects, unique within the process that opened the file system object. The JFN is returned by DosOpen. It is used and an index into the JFN Table to locate the corresponding SFT handle.

A Patricia Tree (PTREE) is a form of tree structure designed to offer a fast look-up facility for generically specified keys. In OS/2 a modified form of the PTREE is use to manage MFTs for fast path-name look-up.
The System Anchor Segment (SAS) is a central system control block used to anchor control blocks for major system components such as:

- File systems
- Device Drivers
- Scheduler
- Memory management

The SAS is built at the beginning of the segment addressable from selector 70 and 78.

Refer to the following for more detailed information:

.A Kernel Debugger and Dump Formatter command.

(No title)

A Master File Table (MFT) entry is used to associate path names with open files (SFTs) and lock records (RLRs). The MFTs are managed in a PTREE structure, which is locatable from the SAS.

See also related structures:

- CDS
- DPB
- SFT
- FSC
- VPB

Refer to the following for more detailed information:

.A Kernel Debugger and Dump Formatter command.
.D Kernel Debugger and Dump Formatter command.

MFT control block format

(No title)

A Record Lock Record (RLR) describes a locked region of a file system record. RLRs are chained from the related and point to the associated SFT. They record the owner of the record lock.

See also related structures:

- CDS
- DPB
- SFT
- FSC
- VPB
A System File Table (SFT) entry is used to describe the attributes of each instance of an open file system object. SFTs are stored in a segment directly locatable from the SAS. SFTs are indirectly locatable from the JFN Table imbedded in the PTDA of each process that opens a file system object.

See also related structures:
- CDS
- DPB
- MFT
- FSC
- VPB

A Job File Number Table (JFT) entry is assigned to each open file system object within a process. The JFT provides a cross-reference to the handle for the corresponding SFT. The JFT is locatable from the PTDA field JFN_pTable (PTDA +0x5b8 (H/R: +0x5b0)) for each process.

The JFT is initially allocated within the PTDA at label JFN_Table (PTDA +0x35e) with 20 entries. If this is expanded by use of the DosSetMaxFH then JFN_pTable is updated to point to the new table.

See also related structures:
- CDS
- DPB
- MFT
- FSC
- VPB

A Volume Parameter Block (VPB) is used to store volume information associated with a file system object. All VPBs are contained within a
single segment locatable from the SAS. Most file system structures contain a VPB handle for an associated volume. The handle is used as an offset into the VPB segment.

See also related structures:

- CDS
- MFT
- SFT
- DBP
- FSC

Refer to the following for more detailed information

. A Kernel Debugger and Dump Formatter command.
. D Kernel Debugger and Dump Formatter command.

VPB control block format

--------------------------------------------

(No title)

A Current Directory Structure (CDS) is used to store file system information about the current directory per drive of each process.

Each CDS is managed in an RMP segment. The PTDA for each process contains an imbedded array of 26 CDS handles, one for each drive. The CDS RMP segment may be located from the SAS.

See also related structures:

- MFT
- SFT
- DPB
- FSC
- VPB

Refer to the following for more detailed information

. A Kernel Debugger and Dump Formatter command.
. D Kernel Debugger and Dump Formatter command.

CDS control block format

--------------------------------------------

(No title)

The Codepage Data Information Block (CDIB) contains country-specific constant information relating to screen, keyboard and printer devices. The CDIB is built from information derived directly from CONFIG.SYS statements.

The CDIB may be located from the SAS.

--------------------------------------------

(No title)
A File System Control Block (FSC) represents an installed file system (IFS). The FSC contains a table of entry points implemented by the file system driver (FSD). All FSCs are located in a single segment whose selector may be obtained from the SAS. See .A command

See also related structures:

- CDS
- MFT
- SFT
- DPB
- VPB

Refer to the following for more detailed information

- FSC control block format

--------------------------------------------

(No title)

A File System Driver (FDS) is a special load module that implements an installed file system (IFS). FSDs are loaded during system initialisation when IFS= statement of CONFIG.SYS is encountered.

Examples of FSDs are:

- HPFS.IFS
- HPFS386.IFS
- CDROM.IFS

--------------------------------------------

(No title)

A Driver Parameter Block (DPB) contains vital information about the state and format of a disk drive. The DPBs are chained together and located in a single segment whose selector may be obtained from the SAS. See .A command

See also related structures:

- CDS
- MFT
- SFT
- FSC
- VPB

Refer to the following for more detailed information

- .A Kernel Debugger and Dump Formatter command.
- .D Kernel Debugger and Dump Formatter command.

--------------------------------------------
Thrashing refers to the state of a system where most of the CPU time is spent paging in and out memory from the swap file. This happens when real storage is heavily over committed and storage references encompass a wide range of virtual pages over a short processing time. Such a condition can indicate a poorly tuned application where paging is caused by the process of accessing data the application needs. A typical scenario is where work data is chained in a single, very extended, queue and no mechanism exits to access the required data without scanning the entire chain. Use of hashing techniques greatly reduce this problem.

The Compatibility Region Mapping Algorithm (also referred to as the thunking algorithm) is used by thunking code to convert 16:16 addresses to 0:32 addresses and vice versa. This is achieved by ensuring LDT selectors have their limits set to 64K so that they tile the compatibility region (0M to 448M). This gives an easy conversion algorithm from the selector:offset address to the 32-bit linear address. In C language syntax this is expressed as follows:

```
linear_address=((selector >> 3) << 16) + offset
selector:offset=((linear_address. >> 13) | 7):(linear_address & 0x0000ffff)
```

Thunking is the process of calling 16-bit code from 32-bit code and vice versa. Thunking consists of applying the CRMA to convert from one form of address to the other and making any stack parameter adjustments either by padding 16-bit operands to 32-bit with leading zeros (16- to 31-bit conversion) or truncating the padded 32-bit value to 16 bits (32- to 16-bit conversion).

The Translation Lookaside Buffer (TLB) is a hardware implemented buffer used for caching linear to physical address mappings. The Intel486(TM) processor provides test registers for manipulating the TLB.

cbargs is the argument count associated with the hardware defined call gate mechanism. The count is the number of words or double-words (as defined by the gate descriptor) that are inserted into a ring 0 stack when ring 2 or ring 3 code executes a call gate instruction.

A paragraph is a unit of memory allocation of 16 bytes. Paragraph aligned allocations lie on a 16-byte boundary.
The Program Data Block is the name given to the DOS PSP by the DEM component of OS/2.

The Program Segment Prefix (PSP) is a DOS control block that forms the header of a loaded program. Under OS/2 the DEM component refers to this as the PDB or Program Data Block.

The Loader Cache is used for saving discardable pages of instance data segments from DLLs loaded from mountable media. The caches is allocated from the kernel heap and has a system object owner ID of cache.

The Scheduler component of OS/2 is responsible for managing threads on queues according to priority and status. Refer to the following for more detailed information:

.P Kernel Debugger and Dump Formatter command.
.PB Kernel Debugger and Dump Formatter command.
.PU Kernel Debugger and Dump Formatter command.
.PQ Kernel Debugger command.

A Screen Group is a logical full screen buffer and keyboard. A number of processes may be assigned to run in a given screen group. The workplace shell is one such screen group. Each screen group is assigned an ID. The screen group assigned to a process is recorded in its Local Information Segment. The currently active screen group is recorded in the Global Information Segment.

Screen Groups are represented by SGCB structures.

Under version 2 of OS/2 the screen group concept has been extended to that of a session.

Refer to the following for information on displaying screen group ids:

.P Kernel Debugger and Dump Formatter command.
The **Screen Group Control Block (SGCB)** is used by the session manager component of the system to represent a **Screen Group**. It contains status information for the screen group and acts as a cross reference between the **Pid** currently associated with a given screen group and **vice versa**.

---

**Sessions** are groups of related processes initiated using **DosStartSession** API. Each session is assigned a logical screen buffer or presentation space. Sessions are identified by a unique ID that corresponds with their **Screen Group ID** (though the range of numbers is extended to include PM sessions, which all share the same screen group).

The following session ID/Screen Group ID ranges are defined:

<table>
<thead>
<tr>
<th>SG Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hard Error Popups</td>
</tr>
<tr>
<td>1</td>
<td>Shell Screen Group</td>
</tr>
<tr>
<td>2</td>
<td>Real Mode Screen Group</td>
</tr>
<tr>
<td>3</td>
<td>VioPopUp Screen Group</td>
</tr>
<tr>
<td>4</td>
<td>First Full Screen Application Session</td>
</tr>
<tr>
<td>15</td>
<td>Last Full Screen Application Session</td>
</tr>
<tr>
<td>16</td>
<td>First Windowable VIO-Session</td>
</tr>
<tr>
<td>31</td>
<td>Last Windowable VIO-Session</td>
</tr>
<tr>
<td>32</td>
<td>First PM session</td>
</tr>
<tr>
<td>255</td>
<td>Last PM session</td>
</tr>
</tbody>
</table>

---

A **process** is a collection of threads that share a common address space.

Each process is primarily represented by a **PTDA** structure and is assigned a unique identifier, the **Pid**.

Processes are organised in hierarchical tree structures known as process or **Command Subtrees**.

---

A **thread** is an independently scheduleable entity that competes for processor resource with other threads.

Each thread is represented by a **TCB** and **TSD** structure.

Threads are organised within processes and assigned a unique identifier within the owning process known as the **Tid**.
All threads within the system are assigned a system wide unique identifier known as the **Thread Slot Number**

Refer to the following for more detailed information

`.P Kernel Debugger and Dump Formatter command.`

```
--------------------------------------------
(No title)

The **Thread Slot Number** is a system wide unique identifier assigned to each thread in the system.

Threads are located from the thread slot table whose linear address is at global symbol:

`_papTCBSlots`

Each slot is a double-word linear address of the corresponding thread's TCB. The first slot (slot=0) is reserved.

Under the Kernel Debugger and Dump Formatter the following symbols may be used to represent particular threads in many of the commands that accept a slot number as a parameter:

* The current or last dispatched thread as recorded in word global variable `_TaskNumber`
# The default thread slot used by the Dump Formatter and Kernel Debugger.

Refer to the following for more detailed information

`.P Kernel Debugger and Dump Formatter command.`

```
--------------------------------------------
(No title)

The **Thread Identifier (tid)** is a value, unique within the owning processes, used to identify the thread. It is not the same as the **Thread Slot Number**, which uniquely identifies a thread, system-wide.

The Tid is used in thread related APIs such as **DosKillThread** and **DosSetPriority**.

```
--------------------------------------------
(No title)

The **Process Identifier (pid)** is a unique system wide value used to identify a given process.

**Note:** It is not the same as the `hptda` which also uniquely identifies a process.

The Pid is used as a handle in process related APIs such as **DosKillProcess** and **DosWaitChild**.

Refer to the following for more detailed information

`.P Kernel Debugger and Dump Formatter command.`

```
--------------------------------------------
(No title)
The Command Subtree Identifier is used to represent a part of a process (or command) tree headed by a particular parent process. The ID used is the Pid of the process that heads the subtree.

Normally a process has a CSID equal to its own Pid. However, when processes become orphaned they acquire the subtree ID of their original parent and become adopted by their grand-parents by acquiring their grand-parents' Pid as their new parent Pid.

Refer to the following for more detailed information

.P Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

A Block Management Package (BMP) is a generalised facility for managing tables of fixed length records. The BMP manages the commitment and decommitment of table storage and maintains a free record list. The table is prefixed with a header structure (the VMBH) which facilitates the management of the BMP.

Typical examples of BMPs are:

- The VMOB table.
- The VMAL table.
- The VMAR table.

--------------------------------------------

(No title)

A Record Management Package (RMP) is a data structure designed for tabulating variable length records. Typically OS/2 uses RMPs to manage:

- Named Storage names
- Open File names
- Directory names
- System Semaphore names

--------------------------------------------

(No title)

DEM is the DOS Emulation component of OS/2.

--------------------------------------------

(No title)

A Page Table Entry (PTE) is a hardware architected structure that is used to map virtual addresses to physical storage addresses.

Refer to the following for more detailed information

.DP Kernel Debugger and Dump Formatter command.
The **User Virtual (UVIRT)** attribute signifies a virtual storage mapping to physical storage made without the full set memory management structures.

The UVIRT attribute may be associated with a number of structures, for example:

- **PTE**
- **LDT** and **GDT** descriptors.
- **VMAL**

In general UVIRT allocations are 'convenience' mappings, which map memory to selector allocation. The attribute is used by Memory Management to signal minimal processing requirements. Typically they are created by device drivers using the DevHip_PhysToUvirt facility. They are also created by the system where a 'quick' form of aliasing is required without the alias being associated with any memory object. Examples of this are:

- VDM private arena memory
- Selectors 5b and 53, 32-bit Code and Data selectors which are used by most 32-bit application programs.
- Selector 30, the TASKAREA selector, which aliases the current process' **TSD**, **TCB** and **PTDA**.

---

**A Virtual Page Structure (VP)** is used by memory management to track the status of a virtual storage frame, whether backed by physical storage, cached by the loader or paged out to the swapper. The Virtual Page Structures are allocated in contiguous storage, anchored from the address specified in global variable: `_pgpVPBase`

Refer to the following for more detailed information

- **MV Kernel Debugger and Dump Formatter command.**
- **VP control block format**

---

**A Page Frame Structure (PF)** is used by page frame management to track the status of a physical storage frame. The Page Frame Structures are allocated in contiguous storage, anchored from the address specified in global variable: `_pft`

Each PF corresponds one to one with a frame of physical storage and provides links to Virtual Page Structures (VPs).

Zero or more PTEs may be pinned to a physical frame, this is reflected in a reference count maintained in the associated PF.

UVIRT mappings have their corresponding PFs reserved unless aliased by non-UVIRT storage.

Refer to the following for more detailed information

- **MP Kernel Debugger and Dump Formatter command.**
The **Global Descriptor Table (GDT)** is a hardware architected control block. The GDT is common to all protect mode processes. It contains descriptors for memory segments common to all protect mode processes.

Refer to the following for more detailed information:

DG Kernel Debugger and Dump Formatter command.

---

The **Local Descriptor Table (LDT)** is a hardware architected table of memory descriptors.

Under OS/2 one LDT is allocated per process except for VDMs in which multiple LDTs are possible. A VDM will allocate one LDT per DPMI client, but if it does not use DPMI then the LDT is not initialised. When multiple LDTs are used in a VDM then the LDTR descriptor contents are updated to make an LDT current. This allows the same LDTR selector may be use regardless of current client.

Refer to the following for more detailed information:

Kernel Debugger and Dump Formatter DL command.

---

A **Virtual Memory Context Record (VMCO)** is used to record the association of shared arena, shared data objects with processes that are using.

Each VMCO is identified by a unique handle referred to as the hco.

Refer to the following for more detailed information

.MC Kernel Debugger and Dump Formatter command.

VMCO control block format

---

The **Virtual Memory Object Record (VMOB)** are used to represent memory objects, that is the instance data associated with a particular virtual address. VMOBs contain pointers to the the owning and requesting objects as well as the corresponding arena record (VMAH).

Each VMOB is identified by a unique handle referred to as the hob.

Refer to the following for more detailed information

.MO Kernel Debugger and Dump Formatter command.

VMOB control block format
The Per-Task Data Area (PTDA) is the anchor point for all process (task) related control information. One PTDA exists per process and from it is located the LDT, TCB chain, Page tables and Arena Headers for a process.

All active PTDAs are addressable, whatever the current process, from a global address in the system arena. However, for the current process an alias address is created using selector 30 and in addition the many of the PTDA field names are declared as public symbols. This allows the fields names in the PTDA for the current process to be referred to directly under the Kernel Debugger and Dump Formatter.

PTDA_Start is the symbol assigned to the beginning of the current PTDA. Using the ? command against this and other PTDA field names allows relative offsets for PTDA fields to be calculated and used in other contexts as offsets from the global PTDA address.

Refer to the following for more detailed information.

PTDA control block format

One Virtual Memory Arena Header Record (VMAH) is allocated per arena to record information about the address range of an arena. The VMAH points to its sentinel arena record (VMAR).

Each VMAH chained in a double linked list.

The system arena VMAH is located at global symbol: _ahvmSys

The shared arena VMAH is located at global symbol: _ahvmShr

For each private arena the VMAH is imbedded in the PTDA at label ptda_ah (PTDA+0x40).

Under OS/2 2.1 the system and shared arena VMAHs are assigned to objects 4 and 5 respectively.

Refer to the following for more detailed information

VMAH control block format

The Virtual Memory Alias Record (VMAL) is used to represent aliased regions of virtual memory. These are either:

    regions of physical storage that may be addressed by more than one virtual, or

    linear address that are not associated with a memory object, such as VDM UVIRT allocations.

When two memory objects are aliases of each other then they need not have coincident sizes or origins within the aliased arena record. Aliases are designed to provide alternative attributes for accessing the same piece of data within or across processes. Compare this with shared instance data within the shared arena, where multiple object records share a common arena record. In this case each object is associated with a unique process and is not considered an alias.
Each **VMAL** is identified by a unique handle referred to as the **hal**.

Refer to the following for more detailed information

- **ML Kernel Debugger command.**
- **VMAL** control block format

--------------------------------------------

**Virtual Memory Kernel Heap** (**VMKH**) structures are used to describe system heap memory. Many objects allocated out of the kernel heap are assigned a **System Object** identifier.

Refer to the following for formats of the kernel heap structures:

- **VMKH** - Kernel Heap Header
- **VMRKH** - Resident Kernel Heap Structures
- **VMSKH** - Swappable Kernel Heap Structures

--------------------------------------------

The **Virtual Memory Arena Record** (**VMAR**) is used to represent a contiguous region of virtual memory allocated in page quantities. Such storage may or may not be committed or resident.

Arena records are chained in a doubly linked lists, one for each arena type. That is, the chain chain exists separately for each private arena, the shared arena and system arena.

Special arena records, known as **Sentinels** head each chain. They describe the entire arena which they head.

All virtual memory is described by by at least one arena record.

Each **VMAR** is identified by a unique handle referred to as the **har**.

Arena also records point to the following related memory structures:

- **VMOB**
- **VMAL**
- **VMCO**

Refer to the following for more detailed information

- **MA Kernel Debugger and Dump Formatter command.**
- **VMAR** control block format

--------------------------------------------

**An Object Table Entry** (**OTE**) describes the address, size and attributes an object within a loaded 32-bit load module.

The corresponding control block for a 16-bit load module is the **STE**.
A Segment Table Entry (STE) describes the address, size and attributes of a segment (object) within a loaded 16-bit load module.

The corresponding control block for a 32-bit load module is the OTE.

The (non-swappable) Module Table Entry (MTE) for a loaded module is used to record information about loaded modules. Since the MTE is allocated in non-swappable only information that must be resident at all times is recorded here. Related information that may be paged out is recorded in its sister control, the Swappable Module Table Entry (SMTE).

The MTE contains the following information:
- pointers to related control blocks such as: SMTE, resource and fix-up tables;
- attributes of the load module;
- Use count for .EXE modules.

Each MTE is identified by a unique handle referred to as the hmte.

The Swappable Module Table Entry (SMTE) contains characteristics of a loaded module that may be paged out of memory. The SMTE is the sister control block to the MTE, which records those characteristics that must be resident at all times.

The SMTE principally contains:
- A pointer to OTE or STE.
- A pointer to the fully qualified module name.
- The entry point and initial stack pointers.

Refer to the following for more detailed information

.LM Kernel Debugger and Dump Formatter command.
The CDA is the Common ABios Data Area. Refer to the following for more detailed information:

.C Kernel Debugger and Dump Formatter command.

--------------------------------------------

The Tread Control Block (TCB) contains per-thread control and status information that must be resident at all times. The swappable counterpart to the TCB is the TSD. Refer to the following for related information:

.P Kernel Debugger and Dump Formatter command.

TCB control block format

--------------------------------------------

The Thread Swappable Data (TSD) control block contains per-thread status and control information that resides in swappable memory and therefore is not required for reference out of context of the related thread. The resident memory counterpart to the TSD is the TCB (Thread Control Block).

The vast majority of the TSD is used as the ring 0 stack when a thread makes a privilege level transition to ring 0 via a call gate descriptor. The base of the ring 0 stack will therefore include the ring 3 call gate stack frame on entry to ring 0 (which is usually kernel or device driver code).

In the debug kernel a dummy page prefixes the used part of the TSD in order to catch ring 0 stack faults. Other information contained in the TSD includes GTD instance data for the corresponding thread's context. This comprises descriptors for:

28: The LDT descriptor.
30: Base selector for ring 0 process instance data, which includes the ring 0 stack, TCBs and PTDA.
38: Floating point emulator instance data
40: FS mapping to the TIB

When an an inter-process thread context switches, descriptors 30 - 40 are loaded into the GDT from the TSD. When an intra-process thread context switches, descriptors 28 - 40 are loaded into the GDT from the TSD.

Refer to the following for related information:

.P Kernel Debugger and Dump Formatter command.

TSD control block format

--------------------------------------------
The memory alias record handle (hal) is an index into the table of memory alias records (VMALs) whose address is located at _parVMAlases.

Refer to the following for more detailed information

.ML Kernel Debugger command.

--------------------------------------------

(No title)

A hco is a handle for a memory context record. This is an index into the table of memory arena records (VMCOs) whose address is located at _pcovmOne.

Refer to the following for more detailed information

.MC Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

The memory arena record handle (har) is an index into the table of memory arena records (VMARs) whose address is located at _parvmOne.

Refer to the following for more detailed information

.MA Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

The memory object record handle (hob) is an index into the table of memory objects records (VMOBs) whose address is located at _pobvmOne

Refer to the following for more detailed information

.MO Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

Block-Ids are conventional tokens used to represent the reason for a thread blocking. This occurs as the result of the kernel entering TKSleep (either directly or via ProcBlock). The address of the block-id is passed to TKSleep and stored in TCBSleepID. A thread wakes when the kernel calls TKWakeup (or ProcRun) with a corresponding block-id. All, zero or the highest priority thread blocked on the block-id will be woken depending on parameter flags. This mechanism is used by most functions and APIs that cause thread execution to be suspended, either for an event or serialisation.

Examples are:

DosSleep
DosSemWait
DosWaitChild
DosRead
DevHlp_ProcBlock

Refer to the following for more detailed information

.PB Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

The PTDA control block handle hptda is the hob of the memory object that contains the PTDA. PTDAs are allocated as pseudo-objects, so do not have Arena Records associated with them. Refer to the following for more detailed information

.MO Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

The MTE control block handle hmte is the hob of the memory object that contains the MTE. MTEs are allocated as pseudo-objects, so do not have Arena Records associated with them. Refer to the following for more detailed information:

.MO Kernel Debugger and Dump Formatter command.
.LM Kernel Debugger and Dump Formatter command.

--------------------------------------------

(No title)

Pseudo-Objects are small system objects that comprise control blocks and other system areas, which for reasons of virtual memory conservation are not represented by a corresponding Arena Records. They are allocated out of the kernel resident heaps and comprise the following types of object:

MTE
VMAH
PTDA
Loader Cache

Refer to the following for more detailed information

.MO Kernel Debugger and Dump Formatter command.

--------------------------------------------