IBM Parallel System Support Programs for AIX Command and Technical Reference

Document Number GC23-3900-02

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xi.

Second Edition (November 1996)

This is a major revision of GC23-3900-01.

This edition applies to Version 2 Release 2 of IBM Parallel System Support Programs for AIX (5765-529), which runs on IBM RS/6000 Scalable POWERparallel Systems, and to all subsequent releases and modifications until otherwise indicated in new editions. Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): KGNVMC(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrcfs@vnet.ibm.com
World Wide Web: http://www.rs6000.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Copyright International Business Machines Corporation 1996. All rights reserved. Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices xi
Trademarks xi
Publicly Available Software xii

About This Book xiii
Who Should Use This Book xiv
How This Book Is Organized xiv
Command Format xiv
Typographic Conventions xv
Obtaining Documentation xv
Related Publications xvi
SP Hardware Publications xvi
SP Software Publications xvi
IBM Virtual Shared Disk Publications xvi
Performance Monitor Publication xvi
LoadLeveler Publications xvii
Parallel Environment for AIX Publications xvii
PVMe for AIX Publications xvii
Client Input Output/Sockets (CLIO/S) Publications xvii
Parallel I/O File System Publications xvii
Network Tape Access and Control System for AIX (NetTAPE) Publications xvii
Other IBM Publications xviii
International Technical Support Organization Publications (Red Books) . . xviii
AIX and RS/6000 Publications xviii
Service xix
Network Queueing System/MVS (NQS/MVS) xix
Network Connectivity xix
Adapters xix
Non-IBM Publications xx
Parallel Computing xx
Tcl xx
Manual Pages for Public Code xxi

Part 1. Command Reference 1

Chapter 1. Commands 3
System Partitioning and Commands 3
add_principal 7
allnimres 9
amq 11
arp 13
cfghsd 15
cfghsdvsd 16
cfgvsd 17
chgcss 19
cksumvsd 21
cmonacct 23
cprdaily 24
create_krb_files 25

© Copyright IBM Corp. 1996

createhsd . . . 27
createvsd . . . 32
crunacct . . . 38
cshutdown . . . 42
CSS_test . . . 48
cstartup . . . 49
ctlhsd . . . 54
ctlvsd . . . 56
defhsd . . . 59
defvsd . . . 61
delnimclient . . . 63
delnimmast . . . 65
dsh . . . 67
dshbak . . . 72
Eannotator . . . 74
Eclock . . . 77
Eduration . . . 82
Efence . . . 84
emconditionctrl script . . . 86
emonctrl script . . . 88
Emonitor daemon . . . 92
Eprimary . . . 94
Equiesce . . . 97
Estart . . . 99
Etopology . . . 102
Eunfence . . . 104
Eunpartition . . . 106
export_clients . . . 107
ext_srvtab . . . 109
ha.vsd . . . 111
hacws_verify . . . 113
haemcfg . . . 115
haemctrl script . . . 119
haemd daemon . . . 125
haemloadcfg . . . 127
haemtrcoff . . . 130
haemtrcon . . . 133
hagsctrl script . . . 136
hagsd daemon . . . 142
hagsglsmd daemon . . . 144
hardmon daemon . . . 146
hats script . . . 150
hatsctrl script . . . 152
hb script . . . 158
hbctrl script . . . 161
hc.vsd . . . 166
hmadm . . . 168
hmcmds . . . 170
hmmon . . . 177
hmreinit . . . 198
hostlist . . . 200
hr script . . . 202
hrctrl script . . . 204
hsdatalst . . . 208

hsdvts . . . 209
ifconfig . . . 210
install_cw . . . 216
install_hacws . . . 217
jm_config . . . 219
jm_install_verify . . . 221
jm_start . . . 223
jm_status . . . 225
jm_stop . . . 228
jm_verify . . . 230
jmcmi_accesscontrol . . . 232
jmcmi_addpool . . . 234
jmcmi_changepool . . . 237
jmcmi_createjmdconfig . . . 240
jmcmi_deletepool . . . 241
jmcmi_servernodes . . . 243
kadmin . . . 245
kadmind daemon . . . 248
kdb_destroy . . . 250
kdb_edit . . . 251
kdb_init . . . 253
kdb_util . . . 254
kdestroy . . . 256
kerberos daemon . . . 257
kinit . . . 259
klist . . . 261
kpasswd . . . 263
kprop . . . 264
kpropd daemon . . . 265
kshd daemon . . . 267
ksrvtgt . . . 270
ksrvutil . . . 271
kstash . . . 273
locate_jm . . . 274
| lppdiff . . . 275
lshacws . . . 279
lshsd . . . 281
lsvsd . . . 283
mkamdent . . . 286
| mkconfig . . . 287
| mkinstall . . . 289
| mknimclient . . . 290
| mknimint . . . 292
| mknimmast . . . 294
| mknimres . . . 296
| ngaddto . . . 298
| ngclean . . . 300
| ngcreate . . . 302
| ngdelete . . . 304
| ngdelfrom . . . 306
| ngfind . . . 308
| nglist . . . 310
| ngnew . . . 312
| ngresolve . . . 314

nodecond . . . 316
nrunacct . . . 318
p_cat . . . 322
pcp . . . 324
pdf . . . 326
penotify . . . 328
perspectives . . . 330
pexec . . . 333
pexscr . . . 335
pfck . . . 336
pfind . . . 338
pfps . . . 340
pls . . . 343
pmanctrl . . . 345
pmandef . . . 349
pmanquery . . . 356
pmanrmdloadSDR . . . 358
pmv . . . 360
ppred . . . 362
pps . . . 364
preparevsd . . . 366
prm . . . 368
psyslclr . . . 370
psyslrpt . . . 372
rcmdtgt . . . 374
rcp . . . 376
removehsd . . . 380
removevsd . . . 382
resumevsd . . . 384
rsh . . . 386
SDR_test . . . 389
SDRAddSyspar . . . 391
SDRArchive . . . 392
SDRChangeAttrValues . . . 393
SDRClearLock . . . 394
SDRCreateAttrs . . . 395
SDRCreateClass . . . 396
SDRCreateFile . . . 397
SDRCreateObjects . . . 398
SDRCreateSystemClass . . . 399
SDRCreateSystemFile . . . 400
SDRDeleteFile . . . 401
SDRDeleteObjects . . . 402
SDRGetObjects . . . 403
SDRListClasses . . . 405
SDRListFiles . . . 406
SDRMoveObjects . . . 407
SDRRemoveSyspar . . . 408
SDRReplaceFile . . . 409
SDRRestore . . . 410
SDRRetrieveFile . . . 412
SDRWhoHasLock . . . 413
seqfile . . . 414
services_config . . . 416

sethacws . . . 417
setup_authent . . . 419
setup_CWS . . . 422
setup_logd . . . 424
setup_server . . . 427
sp_configd . . . 429
sp_configdctrl script . . . 432
spacctnd . . . 437
spacs_cntrl . . . 439
spadaptrs . . . 442
spapply_config . . . 445
spbootins . . . 448
spchuser . . . 453
spcustomize_syspar . . . 455
spcw_addevents . . . 457
spcw_apps . . . 458
spdeladap . . . 460
spdelfram . . . 462
spdelnode . . . 464
spdisplay_config . . . 466
spethernt . . . 470
spevent . . . 473
spframe . . . 476
spget_syspar . . . 478
sphardware . . . 479
sphostnam . . . 482
sphrdwrad . . . 484
splm . . . 487
splogd daemon . . . 491
splst_syspars . . . 494
splst_versions . . . 495
splstdata . . . 497
splsuser . . . 501
spmkuser . . . 502
spmon . . . 504
spmon_ctest . . . 509
spmon_itest . . . 511
spperfmon . . . 513
sprestore_config . . . 516
sprmuser . . . 518
spsitenv . . . 519
spsvrmgr . . . 523
spsyspar . . . 530
spverify_config . . . 533
spvsd . . . 534
startvsd . . . 537
statvsd . . . 539
stopvsd . . . 542
supfilesrv daemon . . . 544
supper . . . 545
suspendvsd . . . 547
sysctl . . . 549
sysctld daemon . . . 552
SYSMAN_test . . . 557

syspar_ctrl . . . 559
sysparaid . . . 564
s1term . . . 569
ucfghsd . . . 571
ucfghsdvsd . . . 572
ucfgvsd . . . 574
unallnimres . . . 576
undefhsd . . . 578
undefvsd . . . 579
updatehsd . . . 580
updatevsdnode . . . 582
updatevsdtab . . . 585
verparvsd . . . 587
vhostname . . . 590
vsdatalst . . . 592
vsddiag . . . 595
vsdelnode . . . 597
vsdelvg . . . 599
vsdnode . . . 601
vsdsklst . . . 604
vsdvg . . . 607
vsdvts . . . 609

Part 2. Technical Reference . . . 611

Chapter 2. SP Files and Other Technical Information . . . 613
haemloadlist file . . . 614
hmacls file . . . 619
.klogin file . . . 621
Kerberos . . . 623
krb.conf file . . . 625
krb.realms file . . . 626
mt_tmcp . . . 627
mte_exits file . . . 632
mte_itable file . . . 638
mte_make file . . . 639
mte_tmpcp . . . 641
SDR_dest_info file . . . 644
sysctl.acl file . . . 645
sysctl.conf file . . . 646

Chapter 3. DTMS Enablement Function I/O Control Requests . . . 649
mt_cassign . . . 650
mt_cskey . . . 653
mt_cupi . . . 655
mt_gdci . . . 658
mt_rdbd . . . 664
mt_rupi . . . 666
mt_srlf . . . 668
mt_srmi . . . 672
mt_srsi . . . 678
mt_srsti . . . 689
mte_cskey . . . 694

mte_cupi 696
mte_eupi 701
mte_mupi 705
mte_rupi 712
mte_srsi 715

Chapter 4. SP Subroutines 725
getvhostname subroutine 726
hacws_set subroutine 728
hacws_stat subroutine 730
setvhostname subroutine 732

Part 3. Appendix 735

Appendix A. Perspectives Colors and Fonts 737
Perspectives Colors with Red, Green, and Blue (RGB) Triplets 737
Perspectives Fonts 741

Glossary of Terms and Abbreviations 743

Index 751

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
AIX/6000
DATABASE 2
ES/9000
ESCON
HACMP/6000
IBM
IBMLink
LoadLeveler
NQS/MVS

© Copyright IBM Corp. 1996

POWERparallel
RS/6000
RS/6000 Scalable POWERparallel Systems
Scalable POWERparallel Systems
SP
System/370
System/390
TURBOWAYS

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Publicly Available Software

This product includes software that is publicly available:

Amd	Berkeley Software Distribution automount daemon
expect	Programmed dialogue with interactive programs
Kerberos	Provides authentication of the execution of remote commands
NTP	Network Time Protocol
Perl	Practical Extraction and Report Language
SUP	Software Update Protocol
Tcl	Tool Command Language
TclX	Tool Command Language Extended
Tk	Tcl-based Tool Kit for X-windows

This book discusses the use of these products only as they apply specifically to the SP system. The distribution for these products includes the source code and associated documentation. (Kerberos does not ship source code.) **/usr/lpp/ssp/public** contains the compressed **tar** files of the publicly available software. (IBM has made minor modifications to the versions of Tcl and Tk used in the SP system to improve their security characteristics. Therefore, the IBM-supplied versions do not match exactly the versions you may build from the compressed **tar** files.) All copyright notices in the documentation must be respected. You can find version and distribution information for each of these products that are part of your selected install options in the **/usr/lpp/ssp/README/ssp.public.README** file.

About This Book

The *IBM Parallel System Support Programs for AIX: Command and Technical Reference* provides detailed syntax and parameter information for all commands you can use to install, customize, and maintain the SP system.

Other books that help you administer and use the SP system include:

- The *IBM Parallel System Support Programs for AIX: System Planning* book helps you plan for the SP system. It tells you how to prepare for software installation and provides considerations for integrating it into your existing computing network.
- The *IBM Parallel System Support Programs for AIX: System Administration Guide* provides information to help you understand and perform SP system administration. It includes concepts and instructions pertaining to:
  - Understanding IBM RS/6000 Scalable POWERparallel Systems (SP)
  - Performing routine system administration tasks
  - Making the system secure
  - Managing sets of nodes
  - Monitoring the system
  - Providing for system availability and problem management
  - Tuning the system
- The *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* helps you understand and perform SP installation, configuration, maintenance, and migration. It includes concepts and instructions for:
  - Installing and configuring the IBM Parallel System Support Programs (PSSP) for AIX
  - Migrating to PSSP Version 2 Release 2
  - Reconfiguring the system
  - Performing system maintenance
- The *IBM Parallel System Support Programs for AIX: Diagnosis and Messages* book provides information to help you diagnose SP system problems and describes error messages, their meaning, and appropriate actions to take to recover from problems.

This book applies to PSSP Version 2 Release 2. To find out what version of PSSP is running on your system, enter the following:

```
SDRGetObjects Syspar syspar_name code_version
```

In response, the system displays something similar to:

```
syspar_name  code_version
Part_1       PSSP-2.2
```

If the response indicates **PSSP-2.2**, this book applies to the version of PSSP that is running on your system.

If you are running mixed levels of PSSP in a system partition, be sure to maintain and refer to the appropriate documentation for whatever versions of PSSP you are running.

© Copyright IBM Corp. 1996

Who Should Use This Book

This book is intended for anyone not familiar with the syntax and use of the SP commands.

How This Book Is Organized

This book consists of three parts. Part 1 of this book is the Command Reference. It contains SP commands which are organized alphabetically. Part 2 of this book is the Technical Reference. It contains the Distributed Tape Management System (DTMS) ioctls and SP files, subroutines, and other technical information. Part 3 of this book is the Appendix. It lists Perspectives colors and fonts.

Command Format

The commands in this book are in the following format:

Purpose Provides the name of the command and a brief description of its purpose.

Syntax Includes a diagram that summarizes the use of the command.

Flags Lists and describes the options that control the behavior of the command.

Operands Lists and describes the objects on which the command operates.

Description Includes a complete description of the command.

Files Lists any SP system files that are read, employed, referred to, or written to by the command, or that are otherwise relevant to its use.

Standard Input
Describes what this command reads from standard input.

Standard Output
Describes what this command writes to standard output.

Standard Error
Describes what and when this command writes to standard error.

Exit Values Describes the values returned and the conditions that caused the values to be returned.

Security Describes who can run this command and provides other security-related information.

Restrictions Lists restrictions beyond the security restrictions described previously.

Implementation Specifics
Identifies the package of each individual command.

Prerequisite Information
Provides a pointer to other documents that would enhance the user's understanding of this command.

Location Specifies the location of the command.

Related Information
Lists SP commands, functions, file formats, and special files that are employed by the command, that have a purpose which is related to that of the command, or that are otherwise of interest within the context of the command. Also listed are related SP documents, other related documents, and miscellaneous information related to the command.

Examples Provides examples of ways in which the command is typically used.

Typographic Conventions

This book uses the following typographic conventions:

Typographic	Usage
Bold	• **Bold** words or characters represent system elements that you must use literally, such as commands, flags, and path names. • **Bold** words also indicate the first use of a term included in the glossary.
Italic	• *Italic* words or characters represent variable values that you must supply. • *Italics* are also used for book titles and for general emphasis in text.
`Constant width`	Examples and information that the system displays appear in `constant width` typeface.
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices. (In other words, it means "or.")
< >	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, <**Enter**> refers to the key on your terminal or workstation that is labeled with the word Enter.
...	An ellipsis indicates that you can repeat the preceding item one or more times.
<**Ctrl-***x*>	The notation <**Ctrl-***x*> indicates a control character sequence. For example, <**Ctrl-c**> means that you hold down the control key while pressing <**c**>.

Obtaining Documentation

This book is available in PostScript format. You can obtain PostScript files for the product documentation by:

- Downloading them from the **lscftp.pok.ibm.com** anonymous FTP server. The files for PSSP can be found in the directory called **/pub/pps/pssp_ftp_prod_dir**.
- Printing the PostScript documentation files that are shipped on the product media.

You can also view or download this book from the IBM RS/6000 web site at **http://www.rs6000.ibm.com**.

Related Publications

SP Hardware Publications

- *IBM RS/6000 Scalable POWERparallel Systems: Site Planning*, GC23-3905
- *IBM RS/6000 Scalable POWERparallel Systems: Maintenance Information, Volume 1, Installation and CE Operations*, GC23-3903
- *IBM RS/6000 Scalable POWERparallel Systems: Maintenance Information, Volume 2, Maintenance Analysis Procedures and Parts Catalog*, GC23-3904
- *IBM RS/6000 Scalable POWERparallel Systems: Library Guide*, GC23-3868
- *IBM RS/6000 Scalable POWERparallel Systems: Installation Aid Licensed Program Specification*, GC23-3838

SP Software Publications

- *IBM Parallel System Support Programs for AIX: System Planning*, GC23-3902
- *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897
- *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898
- *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*, GC23-3899
- *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*, SC23-3996
- *IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference*, SC28-1675
- *IBM Parallel System Support Programs for AIX: Licensed Program Specification*, GC23-3901

As an alternative to ordering the individual books, you can use SBOF-8587 to order the entire SP software library.

IBM Virtual Shared Disk Publications

- *IBM Parallel System Support Programs for AIX: Managing Shared Disks: IBM Virtual Shared Disks, Data Striping Devices, and the IBM Recoverable Virtual Shared Disk Licensed Program*, SC23-3849
- *IBM Recoverable Virtual Shared Disk: Licensed Program Specification*, GC23-3848

Performance Monitor Publication

- *IBM Performance Toolbox Parallel Extensions for AIX: Guide and Reference*, SC23-3997

LoadLeveler Publications

- *Using and Administering LoadLeveler*, SC23-3989
- *IBM LoadLeveler: Licensed Program Specification*, GH23-0040

Parallel Environment for AIX Publications

- *IBM Parallel Environment for AIX: Hitchhiker's Guide*, GC23-3895
- *IBM Parallel Environment for AIX: Operation and Use, Volume 1*, SC28-1979
- *IBM Parallel Environment for AIX: Operation and Use, Volume 2*, SC28-1980
- *IBM Parallel Environment for AIX: Installation Guide*, GC28-1981
- *IBM Parallel Environment for AIX: Messages*, GC28-1982
- *IBM Parallel Environment for AIX: MPL Programming and Subroutine Reference*, GC23-3893
- *IBM Parallel Environment for AIX: MPI Programming and Subroutine Reference*, GC23-3894
- *IBM Parallel Environment for AIX: Licensed Program Specification*, GC23-3896

As an alternative to ordering the individual books, you can use SBOF-8588 to order the entire Parallel Environment for AIX library.

PVMe for AIX Publications

- *IBM PVMe for AIX: Fact Sheet*, GC23-3907
- *IBM PVMe for AIX: User's Guide and Reference*, GC23-3884
- *IBM PVMe for AIX: Licensed Program Specification*, GC23-3885

Client Input Output/Sockets (CLIO/S) Publications

- *IBM CLIO/S General Information*, GC23-3879
- *IBM CLIO/S Licensed Program Specification*, GC23-3789
- *IBM CLIO/S User's Guide and Reference*, GC28-1676

Parallel I/O File System Publications

- *IBM AIX Parallel I/O File System: Fact Sheet*, G325-0649
- *IBM AIX Parallel I/O File System: Installation, Administration, and Use*, SH34-6065
- *IBM AIX Parallel I/O File System: Licensed Program Specification*, GH34-6066

Network Tape Access and Control System for AIX (NetTAPE) Publications

- *IBM NetTAPE General Information*, GC23-3990
- *IBM NetTAPE User's Guide and Reference*, available from your IBM representative

Other IBM Publications

International Technical Support Organization Publications (Red Books)

- *IBM International Technical Support Centers Implementing High Availability on RS/6000 SP*, SG24-4742
- *IBM International Technical Support Centers RS/6000 SP High Availability Infrastructure*, SG24-4838
- *IBM International Technical Support Centers RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868

AIX and RS/6000 Publications

- *IBM AIX Version 4 Getting Started*, SC23-2527
- *IBM General Concepts and Procedures for RS/6000*, GC23-2202
- *IBM AIX Version 4 Files Reference*, SC23-2512
- *IBM AIX Version 4 System Management Guide: Communications and Networks*, SC23-2526
- *IBM AIX Version 4.1 Installation Guide*, SC23-2550
- *IBM AIX Version 4.2 Installation Guide*, SC23-1924
- *IBM AIX Version 4 Commands Reference*, SBOF-1851
- *IBM AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365
- *IBM AIX Version 4 Problem Solving Guide and Reference*, SC23-2606
- *IBM AIX Version 4 Messages Guide and Reference*, SC23-2641
- *IBM AIX Version 4.1 Network Installation Management Guide and Reference*, SC23-2627
- *IBM AIX Version 4.2 Network Installation Management Guide and Reference*, SC23-1926
- *IBM AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525
- *IBM AIX Version 4 General Programming Concepts: Writing and Debugging Programs*, SC23-2533
- *IBM AIX Version 4 Communications Programming Concepts*, SC23-2610
- *Diskless Workstation Management Guide*, SC23-2433
- *C++ for AIX/6000: Language Reference*, SC09-1606
- *C++ for AIX/6000: Standard Class Library Reference*, SC09-1604
- *C++ for AIX/6000: User's Guide*, SC09-1605
- *IBM Performance Toolbox 1.2 and 2 for AIX: Guide and Reference*, SC23-2625

Service

- *RS/6000 Problem Solving Guide*, SC23-2204
- *Common Diagnostics Micro Channel*, SA23-2765
- *7012 300 Series Operator Guide*, SA23-2623
- *7012 300 Series Installation and Service Guide*, SA23-2624
- *7013 500 Series Operator Guide*, SA23-2621
- *7013 500 Series Installation and Service Guide*, SA23-2622
- *Electrical Safety for IBM Service Representatives*, S229-8124
- *Service Information Binder*, SX33-6060
- *7015 Model R30 CPU Operator Guide*, SA23-2742
- *7015 Model R30 CPU Service Guide*, SA23-2743
- *Supplemental Information for 7012 G Series Models, 7013 J Series Models, 7015 Models R30 and R40* (No order number — this book comes as part of the ship group with the SP hardware)

Network Queueing System/MVS (NQS/MVS)

- *IBM NQS/MVS Installing and Administering*, SC23-0232
- *IBM NQS/MVS Client User's Guide*, SC23-0231
- *IBM NQS/MVS General Information*, GC23-3684
- *IBM NQS/MVS Licensed Program Specifications*, GC23-0233

Network Connectivity

- *IBM LAN Cabling System Planning and Installation Guide*, GA27-3361
- *IBM Cabling System Optical Fiber Planning and Installation Guide*, GA27-3943
- *IBM 8250/8260/8285 Planning and Site Preparation Guide*, GA33-0285
- *IBM 6611 Network Processor: Introduction and Planning Guide*, GK2T-0334

Adapters

- *FDDI Introduction and Planning Guide*, GA27-3892
- *FDDI User's Guide and Programming Reference*, SC28-2823
- *Block Multiplexer Channel Adapter User's Guide and Service Information*, SC23-2427
- *Planning for Fiber Optic Channel Links*, GA23-0367
- *Token-Ring Network Introduction and Planning Guide*, GA27-3677
- *RS/6000 Token Ring Adapter Card*, G511-1681
- *HiPPI User's Guide and Programmer's Reference*, SA23-0369 and SA23-2488
- *AIX Parallel and ESCON Channel Tape Attachment/6000 Installation and User's Guide*, GA32-0311
- *9334 SCSI Expansion Units Operator Guide*, GA33-3232

- *9334 SCSI Model 010 Expansion Unit: Installation and Service Guide*, SY33-0165
- *9334 SCSI Model 500 Expansion Unit: Installation and Service Guide*, SY33-0167
- *SCSI-2 Fast/Wide Adapter*, SC23-2646
- *IBM SCSI-2 Fast/Wide Adapter/A Technical Reference*, S83G-7545
- *Turboways 100 User's Guide ATM*, GA27-4057
- *9333 Model 010 and 011 High-Performance Disk-Drive Subsystem Operator Guide*, GA33-3208
- *9333 Model 010 and 011 High-Performance Disk-Drive Subsystem Installation and Service Guide*, SY33-0161
- *9333 Model 010 and 011 High-Performance Disk-Drive Subsystem Hardware Technical Information*, SA33-3209
- *9333 Model 500 and 501 High-Performance Disk-Drive Subsystem Operator Guide*, GA33-3234
- *9333 Model 500 and 501 High-Performance Disk-Drive Subsystem Installation and Service Guide*, SY33-0168
- *9333 Model 500 and 501 High-Performance Disk-Drive Subsystem Hardware Technical Information*, SA33-3235
- *IBM SCSI Tape Drive, Medium Changer, and Library Device Drivers Installation and User's Guide*, GC35-0154

Non-IBM Publications

Parallel Computing

- Almasi, G., Gottlieb, A., *Highly Parallel Computing*, Benjamin-Cummings Publishing Company, Inc., 1989.
- Gropp, W., Lusk, E., Skjellum, A., *Using MPI*, The MIT Press, 1994.
- Message Passing Interface Forum, MPI: *A Message-Passing Interface Standard*. Version 1.1, University of Tennessee, Knoxville, Tennessee, June 6, 1995.
- Foster, I., *Designing and Building Parallel Programs*. Addison-Wesley, 1995.
- Bergmark, D., Pottle, M. *Optimization and Parallelization of a Commodity Trade Model for the SP1*. Cornell Theory Center, Cornell University, June, 1994.
- Phister, Gregory, F., *In Search of Clusters*. Prentice Hall, 1995.

Tcl

- Ousterhout, John K., *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, MA, 1994, ISBN 0-201-63337-X.

Manual Pages for Public Code

The following manual pages for public code are available in this product:

SUP	/usr/lpp/ssp/man/man1/sup.1
Amd	/usr/lpp/ssp/man/man8/amd.8
	/usr/lpp/ssp/man/man8/amq.8
NTP	/usr/lpp/ssp/man/man8/xntpd.8
	/usr/lpp/ssp/man/man8/xntpdc.8
Perl (Version 4.036)	/usr/lpp/ssp/perl/man/perl.man
	/usr/lpp/ssp/perl/man/h2ph.man
	/usr/lpp/ssp/perl/man/s2p.man
	/usr/lpp/ssp/perl/man/a2p.man
Perl (Version 5.002)	Man pages are in the /usr/lpp/ssp/perl5/man/man1 directory

Manual pages and other documentation for Tcl, TclX, Tk, and expect can be found in the compressed **tar** files located in **/usr/lpp/ssp/public**.

Part 1. Command Reference

© Copyright IBM Corp. 1996

Chapter 1. Commands

This part of the book contains the SP commands.

To access the SP online manual pages, set the MANPATH environment variable as follows:

for ksh

```
export MANPATH=$MANPATH:/usr/lpp/ssp/man
```

for csh

```
setenv MANPATH $MANPATH\:/usr/lpp/ssp/man
```

System Partitioning and Commands

When you partition your system, you create one or more system partitions which, for most tasks, function as separate and distinct logical SP systems. Most commands function within the boundary of the system partition in which they are executed. A number of commands, however, continue to treat the SP as a single entity and do not respect system partition boundaries. That is, in their normal function they may affect a node or other entity outside of the current system partition. In addition, some commands which normally function only within the current system partition have been given a new parameter which, when used, allows the scope of that command to exceed the boundaries of the current system partition.

The following table summarizes those commands which can exceed the boundary of the current system partition. Unless otherwise stated, commands not listed in this table have as their scope the current system partition.

Command	Effect
arp	Can reference any node (by its host name) in any system partition.
Amd commands	Host names need not be in the current system partition.
crunacct	Merges accounting data from all nodes regardless of system partition boundaries.
cshutdown -G	The **-G** flag allows specification of target nodes outside of the current system partition.
cstartup -G	The **-G** flag allows specification of target nodes outside of the current system partition.
dsh **dsh -w** {*hostname* | -}	Hosts added to the working collective by host name need not be in the current system partition.
dsh -aG	The **-G** flag modifies the **-a** flag (all nodes in the current system partition) by expanding the scope to all nodes in the entire physical SP system.
Eclock	There is a single switch clock for the SP regardless of the number of system partitions.
Efence -G	The **-G** flag allows specification of nodes outside of the current system partition.
emonctrl -c	The system partition-sensitive control script for the **emon** subsystem supports the **-c** option, which crosses system partitions.
Eunfence -G	The **-G** flag allows specification of nodes outside of the current system partition.
haemctrl -c	The system partition-sensitive control script for the **haem** subsystem supports the **-c** option, which crosses system partitions.

© Copyright IBM Corp. 1996

Command	Effect
hagsctrl -c	The system partition-sensitive control script for the **hags** subsystem supports the **-c** option, which crosses system partitions.
hatsctrl -c	The system partition-sensitive control script for the **hats** subsystem supports the **-c** option, which crosses system partitions.
hbctrl -c	The system partition-sensitive control script for the **hb** subsystem supports the **-c** option, which crosses system partitions.
hmcmds -G	The **-G** flag allows the **hmcmds** commands to be sent to any hardware on the SP system.
hmmon -G	The **-G** flag allows for the specification of hardware outside of the current system partition.
hostlist **hostlist -f** *filename* **hostlist -w** *hostname*	Host names need not be in the current system partition.
hostlist -aG | **-nG** | **-sG**	The **-G** flag modifies the **-a**, **-n**, or **-s** flag by expanding the scope to the entire physical SP system.
hrctrl -c	The system partition-sensitive control script for the **hr** subsystem supports the **-c** option, which crosses system partitions.
hsdatalst -G	The **-G** flag causes the display of HSD information to be for all system partitions.
lppdiff -aG	The **-G** flag modifies the **-a** flag (all nodes in the current system partition) by expanding the scope to all nodes in the entire physical SP system.
nodecond -G	The **-G** flag allows specification of a node outside of the current system partition.
psyslrpt -w *hostnames*	The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition).
psyslclr -w *hostnames*	The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition).
penotify -w *hostnames*	The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition).
pmanctrl -c	The system partition-sensitive control script for the **pman** subsystem supports the **-c** option, which crosses system partitions.
Parallel commands: • **p_cat** • **pcp** • **pdf** • **pfck** • **pexec** • **pexscr** • **pfind** • **pfps** • **pls** • **pmv** • **ppred** • **pps** • **prm**	Parallel commands can take the following options and will behave accordingly: **-w** Host names specified with **-w** need not be in the current system partition. *noderange* Nodes specified by *noderange* must be in the current system partition. *hostlist_args* Host names specified with **hostlist** options **-w** or **-G** need not be in the current system partition (any other **hostlist** options operate within the current system partition).
SDRArchive, **SDRRestore**	Archives/restores the SDR representing the entire SP.
SDRGetObjects -G	The **-G** flag allows for retrieval of partitioned class objects from partitions other than the current system partition. Without the **-G**, objects which are in a partitioned class are retrieved from the current system partition only.
SDRMoveObjects	Moves objects from one system partition to another.

Command	Effect
Other SDR commands	SDR commands that create, change or delete values work within the system partition. Note though that System classes (Frame, for example) are shared among all system partitions. Changes to system classes will affect other system partitions.
Security commands: • **ext_srvtab** • **kadmin** • **kdb_destroy** • **kdb_edit** • **kdb_init** • **kdb_util** • **kdestroy** • **kinit** • **klist** • **kpasswd** • **kprop** • **ksrvtgt** • **ksrvutil** • **kstash** • **rcmdtgt** • **rcp** • **rsh** • **setup_authent**	The function of these security commands is unchanged under system partitioning. That is, if they previously affected the entire SP, they continue to do so even if the system has been partitioned. If they previously had the ability to affect a remote node (**rsh**, **rcp**, for example), that function is unchanged in a system partitioned environment.
spapply_config	Applies a system partition configuration to the entire SP.
spbootins	If a boot server outside of the current system partition is specified, that node is prepared appropriately.
sp_configdctrl -c	The system partition-sensitive control script for the **sp_configd** subsystem supports the **-c** option, which crosses system partitions.
spdelfram -G	The **-G** flag allows the deletion of a frame containing nodes outside of the current system partition.
spframe	Configures data for one or more frames across the entire SP.
splm	The target nodes defined in the input table can include nodes from any system partition.
splst_versions -G	The **-G** flag allows for retrieval of PSSP version information from nodes outside the current system partition.
splstdata -G	The **-G** flag allows display of information on nodes and adapters outside of the current system partition.
spmon -G	The **-G** flag allows specification of nodes outside of the current system partition. The **-G** flag is required when performing operations on any frame or switch.
sprestore_config	Restores the entire SP SDR from a previously made archive.
spsitenv	Site environment variables are specified for the SP system as a whole. The specification of **acct_master=** can be any node in the SP regardless of system partition. The specification of **install_image=** may cause boot server nodes outside of the current system partition to refresh the default installation image they will serve to their nodes.
spverify_config	Verifies the configuration of all system partitions in the SP system.
supper	File collections are implemented and managed without respect to system partition boundaries.
sysctl	The Sysctl client can send requests to any node in the SP.
syspar_ctrl -c -G	The **-c** and **-G** flags allow for the crossing of system partitions in providing a single interface to the control scripts for the system partition-sensitive subsystems.

Command	Effect
s1term -G	The **-G** flag allows specification of a node outside of the current system partition.
vsdatalst -G	The **-G** flag causes the display of IBM Virtual Shared Disk information to be for all system partitions.
vsdsklst -G	The **-G** flag specifies the display of information for disks outside the current system partition.

add_principal

Purpose

add_principal – Creates principals in the authentication database.

Syntax

add_principal [–**r** *realm_name*] [–**v**] *file_name*

Flags

–**r** Adds principals to a realm other than the local realm.

–**v** Specifies verbose mode. A message is written to standard output for each principal added to the authentication database.

Operands

file_name Specifies the file containing principal names and passwords to add to the authentication database.

Description

This command provides an interface to the authentication database to add an entry for a user or service instance, supplying the password used to generate the encrypted private key. The **add_principal** command is suitable for mass addition of users or multiple instances of servers (for example, SP nodes).

This command operates noninteractively if you have a valid ticket-granting-ticket (TGT) for your **admin** instance in the applicable realm. A TGT can be obtained using the **kinit** command. If you do not have a TGT for the **admin** instance for the realm in which you are adding principals, or if the **add_principal** command cannot obtain a service ticket for changing passwords using the **admin** TGT, the user is prompted for the password for the user's **admin** instance.

Administrators use the **add_principal** command to register new users and services instances to the authentication database. An administrator must have a principal ID with an instance of **admin**. Also, *user_name***.admin** must appear in the **admin_acl.add** Access Control List (ACL).

The **add_principal** program communicates over the network with the **kadmind** program, which runs on the machine housing the primary authentication database. The **kadmind** program creates new entries in the database using data provided by this command.

When using the **add_principal** command, the principal's expiration date and maximum ticket lifetime are set to the default values. To override the defaults, the root user must use the **kdb_edit** command to modify those attributes.

Input to the program is read from the file specified by the *file_name* argument. It contains one line of information for each principal to be added, in the following format:

```
name[.instance][@realm] password
```

Note: The @realm cannot be different from the local realm or the realm argument if the –**r** option is specified.

For user entries with a NULL instance, this format matches that of the log file created by the **spmkuser** command. Any form of white space can surround the two fields. Blank lines are ignored. Any line containing a # as the first nonwhite space character, is treated as a comment.

Since the input file contains principal identifiers and their passwords, ensure that access to the file is controlled. You should remove the input file containing the unencrypted passwords after using it, or delete the passwords from it.

The **add_principal** command does not add principals to an AFS authentication database. If authentication services are provided through AFS, use the AFS **kas** command to add principals to the database. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for an overview.

Files

/var/kerberos/database/admin_acl.add
Access Control List file.

Exit Values

0 Indicates success. It does not mean that all IDs were added. Individual messages indicate what was added.

nonzero Indicates a failure with an appropriate message.

Related Information

Commands: **kadmin**, **kinit**, **kpasswd**, **ksrvutil**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

allnimres

Purpose

allnimres – Allocates Network Installation Management (NIM) resources from a NIM master to a NIM client.

Syntax

allnimres –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to which to allocate installation resources. The *node_list* is a comma-separated list of node numbers.

Operands

None.

Description

Use this command to allocate all necessary NIM resources to a client based on the client's **bootp_response** in the System Data Repository (SDR). This includes executing the **bos_inst** command for allocation of the boot resource and nimscript resource. At the end of this command, nodes are ready to netboot to run installation, diagnostics, or maintenance. If the node's **bootp_response** is “disk”, all NIM resources are deallocated from the node.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

allnimres

Location

/usr/lpp/ssp/bin/allnimres

Related Information

Commands: **setup_server**, **unallnimres**

Examples

To allocate boot/installation resources to boot/install client nodes 1, 3, and 5 from their respective boot/install servers, enter:

```
allnimres -l 1,3,5
```

amq

Purpose

amq – Provides information about the Amd mounts.

Syntax

amq [*mount_points*] [–**f**] [–**h** *host*] [–**m**] [–**M** *map_entry*] [–**s**] [–**u** *directory*] [–**v**] [–**l** *log_file* | –**l syslog**] [–**x** *log_flags*] [–**D** *dbg_opts*]

Flags

–**f** Flushes the internal mount map cache.

–**h** *host* Queries Amd on the specified host rather than the local host.

–**m** Displays information about the mounted file systems rather than the automount points. This information includes: the mount information, the mount point, the file system type, the number of references to this file system, the server host name, the state of the file server, any errors that occurred.

–**M** *map_entry* Supplies Amd with a new map entry to mount. Amd evaluates if the request is from a privileged UDP port on the local machine before mounting the map.

–**s** Displays global statistics to include the number of: requests deferred, stale filehandles, mounts that were OK or failed, unmounts that failed.

–**u** *directory* Safely unmounts an automounted file system by expiring the time-to-live interval.

–**v** Displays the version of Amd that is running.

–**l** Choice of *log_file* | **syslog**. Specifies the form of logging, where *log_file* can be /**dev**/**stderr** (for standard error) or any other string to specify a file to use for the log. If you specify **syslog**, Amd uses the AIX **syslog** mechanism. (This is lowercase **l**, as in **l**ist.)

–**x** *log_flags* Specifies the type of log messages to produce, where *log_flags* is a comma-separated list selected from the following:

fatal	Fatal errors
error	Nonfatal errors
user	Nonfatal user errors
warn	Recoverable errors
info	Information messages
map	Mount map usage
stats	Additional statistics
all	All of the above

–**D** *dbg_opts* Specifies the type of debugging trace. This is only available if Amd was compiled with the –DDEBUG option. Two common values for *dbg_opts* are **trace** and **test**. See the program source for more information.

Operands

mount_points Mount point from which you execute this script if the control workstation is not being used as a file server.

Description

When you run **amq** with no mount point or flags, it produces a brief list of all existing mount points created by Amd.

When you run **amq** and specify mount points without any other flags, it produces statistics for the volume name.

Examples

To produce the default list of all existing mount points, enter:

```
[tserv11][/]> /etc/amd/amq
/           root    "root"               tserv5:(pid12647)
/u          toplvl  amd.u                /u
/u/ptest1   link    ./home/peggy/ptest1  /home/peggy/ptest1
```

To produce the list global statistics, enter:

```
[tserv11][/]> /etc/amd/amq -s
requests  stale     mount     mount     unmount
deferred  fhandles  ok        failed    failed
0         0         2         0         0
```

To check another host, enter:

```
[tserv11][/]> /etc/amd/amq -h tserv11
   /   root    "root"  tserv11:(pid1902)
   /u  toplvl  amd.u   /u
```

arp

Purpose

/usr/lpp/ssp/css/arp – Displays and modifies address resolution.

Syntax

arp {*host_name* | –**a** [/**dev**/**kmem**]} | –**d** *host_name* |
–**s** *type host_name adapter_address* [*route*] [*temp*] [*pub*] |
–**f** *file_name* [*type*]

Parameters

–**a** Displays all of the current Address Resolution Protocol (ARP) entries. Use the **crash** command to look at KMEM or UMUnix variables. Specify the –**a** /**dev**/**kmem** flag to display ARP information for kernel memory.

–**d** *host_name*
Deletes an ARP entry for the host specified by the *host_name* variable if the user has root user authority.

–**f** *file_name*
Causes the file specified by the *file_name* variable to be read and multiple entries to be set in the ARP tables. Entries in the file should be in the form:

```
type host_name adapter_address [route] [temp] [pub]
```

–**s** *type host_name adapter_address* **[***route***] [***temp***] [***pub***]**
Creates an ARP entry for the host specified by the *host_name* variable with the adapter address specified by the *adapter_address* variable. The adapter address is given as 6 hexadecimal bytes separated by colons. The line must be in the following format:

```
type host_name adapter_address [route] [temp] [pub]
```

where:

type Specifies the type of hardware address as follows:

ether	An Ethernet interface
802.3	An 802.3 interface
switch	A Scalable POWERparallel Switch (SP Switch) or a High Performance Switch
fddi	A Fiber Distributed Data Interface
802.5	A token-ring interface

host_name Specifies the host_name for which to create an entry.

adapter_address
Specifies the physical address (switch node number) for the switch adapters.

route Specifies the route for a token-ring interface or Fiber Distributed Data Interface (FDDI) as defined in the token-ring or FDDI header.

temp Specifies that this ARP table entry is temporary. The table entry is permanent if this argument is omitted.

pub Specifies that this table entry is to be published, and that this system acts as an ARP server responding to requests for *host_name*, even though the host address is not its own.

Description

The **arp** command has been modified to add support for the switch. This command is valid only on an SP system.

The **arp** command displays and modifies the Internet-to-adapter address translation tables used by ARP. The **arp** command displays the current ARP entry for the host specified by the *host_name* variable. The host can be specified by name or number, using Internet dotted decimal notation.

Related Information

SP Command: **ifconfig**

AIX Commands: **crash**, **netstat**

AIX Daemon: **inetd**

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the SP Switch and the High Performance Switch.

Refer to "TCP/IP Protocols" in *AIX Version 4.1 System Management Guide: Communications and Networks*.

Examples

1. To add a single entry to the **arp** mapping tables until the next time the system is restarted, enter:

   ```
   arp -s switch host2 1
   ```

2. To delete a map table entry for the specified host with the **arp** command, enter:

   ```
   arp -d host1
   ```

cfghsd

Purpose

cfghsd – Configures a data striping device (HSD) for an IBM Virtual Shared Disk.

Syntax

cfghsd –**a** *hsd_name* ...

Flags

–**a** Specifies all the data striping devices that have been defined.

Operands

hsd_name Specifies a defined HSD. All underlying IBM Virtual Shared Disks in the HSD must be configured before using this command.

Description

This command configures the already defined HSDs and makes them available. The command extracts information from the System Data Repository (SDR).

Files

/usr/lpp/csd/bin/cfghsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defhsd**, **hsdatalst**, **lshsd**, **ucfghsd**

Examples

To make the HSD *hsd1* available, enter:

```
cfghsd hsd1
```

cfghsdvsd

Purpose

cfghsdvsd – Configures a data striping device (HSD) and the IBM Virtual Shared Disks that comprise it on one node.

Syntax

cfghsdvsd –**a** | {*hsd_name*...}

Flags

–**a** Specifies that all the data striping devices defined on this system or system partition are to be configured (made available).

Operands

hsd_name Specifies the names of defined HSDs that are to be configured. This command configures the underlying IBM Virtual Shared Disks as well.

Description

Use this command to configure already-defined HSDs and their underlying IBM Virtual Shared Disks and make them available.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit hsd_mgmt
```

and select the Configure an HSD and its underlying IBM Virtual Shared Disks option.

Files

/usr/lpp/csd/bin/cfghsdvsd Specifies the command file.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsd**, **cfgvsd**, **ucfghsdvsd**

Examples

To configure the data striping device *hsd1* and the IBM Virtual Shared Disks that comprise it, enter:

```
cfghsdvsd hsd1
```

cfgvsd

Purpose

cfgvsd – Configures an IBM Virtual Shared Disk.

Syntax

cfgvsd –**a** | *vsd_name* ...

Flags

–**a** Specifies all IBM Virtual Shared Disks that have been defined.

Operands

vsd_name Specifies a defined IBM Virtual Shared Disk.

Description

Use this command to configure the already defined IBM Virtual Shared Disks and bring them to the stopped state. It does not make the IBM Virtual Shared Disk available. The command extracts information from the System Data Repository (SDR).

You can use the System Management Interface Tool (SMIT) to run the **cfgvsd** command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Configure an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/cfgvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Examples

To bring the IBM Virtual Shared Disk **vsd1vg1n1** from the defined state to the stopped state, enter:

```
cfgvsd vsd1vg1n1
```

chgcss

Purpose

chgcss – Applies configuration changes to a Scalable POWERparallel Switch (SP Switch) Communications Adapter (Type 6-9) or a High Performance Switch Communications Adapter Type 2.

Implementation Note

Configuration changes are later applied to the device when it is configured at system reboot.

Syntax

chgcss –**l** *name* –**a** *attr=value* [–**a** *attr=value*]

Flags

–**l** *name* Specifies the device logical name in the Customized Devices object class whose attribute values should be changed.

–**a** *attr=value*
Identifies an attribute to be changed and the value to which it should be changed

where:

attr Specifies the IP buffer pool for the switch device driver as follows:

rpoolsize IP receive buffer pool

spoolsize IP send buffer pool

value Specifies the IP buffer pool size in bytes.

Operands

None.

Description

Use this command to apply configuration changes to an SP Switch Communications Adapter (Type 6-9) or a High Performance Switch Communications Adapter Type 2.

Security

You must have root privilege to run this command.

Prerequisite Information

For additional information on values for the **rpoolsize** and **spoolsize** attributes, refer to the "Tuning the SP System" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Related Information

AIX Command: **lsattr**

Examples

1. To change the size of the IP receive buffer to 1024K, enter:

   ```
   chgcss -l css0 -a rpoolsize=0x100000
   ```

2. To change the size of the IP send and receive buffers to 1024K, enter:

   ```
   chgcss -l css0 -a rpoolsize=1048576 -a spoolsize=1048576
   ```

cksumvsd

Purpose

cksumvsd – Views and manipulates the IBM Virtual Shared Disk checksum parameters.

Syntax

cksumvsd [–**s**] [–**R**] [–**i** | –**l**]

Flags

–**s** Shows IP checksum counters only.

–**R** Resets IP checksum counters.

–**i** Calculates IP checksum on all IBM Virtual Shared Disk remote messages.

–**l** Indicates not to calculate IP checksum on all IBM Virtual Shared Disk remote messages.

If no flags are specified, the current setting of all IBM Virtual Shared Disk checksum parameters and counters are displayed.

Operands

None.

Description

The IBM Virtual Shared Disk IP device driver can calculate and send checksums on remote packets it sends. It also can calculate and verify checksums on remote packets it receives. The **cksumvsd** command is used to tell the device driver whether to perform checksum processing. The default is no checksumming.

Issuing **cksumvsd** –**i** turns on checksumming on the node on which it is run. **cksumvsd** –**i** must be issued on all IBM Virtual Shared Disk nodes in the system partition, or the IBM Virtual Shared Disk software will stop working properly on the system partition. If node A has **cksumvsd** –**i** (checksumming turned on) and node B has **cksumvsd** –**l** (checksumming turned off, the default), then A will reject all messages from B (both requests and replies), since A's checksum verification will fail on all B's messages. The safe way to run **cksumvsd** –**i** is to make sure that all IBM Virtual Shared Disks on all nodes are in the STOPPED or SUSPENDED states, issue **cksumvsd** –**i** on all nodes, then resume the needed IBM Virtual Shared Disks on all nodes.

In checksumming mode, the IBM Virtual Shared Disk IP device driver keeps a counter of the number of packets received with good checksums, and the number received with bad checksums. **cksumvsd** and **statvsd** both display these values (**statvsd** calls **cksumvsd** –**s**).

cksumvsd dynamically responds to the configuration of the IBM Virtual Shared Disk IP device driver loaded in the kernel. Its output and function may change if the IBM Virtual Shared Disk IP device driver configuration changes.

Files

/dev/kmem **cksumvsd** reads and writes **/dev/kmem** to get information to and from the IBM Virtual Shared Disk IP device driver in the kernel.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Command: **cfgvsd**

Examples

1. To display the IBM Virtual Shared Disk checksum settings and counter values, enter:

   ```
   cksumvsd
   ```

   You should receive output similar to the following:

   ```
   VSD cksum: current values:
   do_ip_checksum: 0
   ipcksum_cntr:   350 good,       0 bad,  0 % bad.
   ```

   The IBM Virtual Shared Disk checksumming is currently turned off on the node. Prior to this, checksumming was turned on and 350 IBM Virtual Shared Disk remote messages were received, all with good checksumming.

2. To turn IBM Virtual Shared Disk checksumming on and display counters, enter:

   ```
   cksumvsd -i
   ```

   You should receive output similar to the following:

   ```
   VSD cksum: current values:
   do_ip_checksum: 0
   ipcksum_cntr:   350 good,       0 bad,  0 % bad.
   VSD cksum: new values:
   do_ip_checksum: 1
   ipcksum_cntr:   350 good,       0 bad,  0 % bad.
   ```

   The command displays old and new values. As before, the node has received 350 IBM Virtual Shared Disk remote messages with good checksums.

3. To display only the IBM Virtual Shared Disk checksum counters, enter:

   ```
   cksumvsd -s
   ```

   You should receive output similar to the following:

   ```
   ipcksum_cntr:   350 good,       0 bad,  0 % bad.
   ```

cmonacct

Purpose

cmonacct – Performs monthly or periodic SP accounting.

Syntax

cmonacct [*number*]

Flags

None.

Operands

number Specifies which month or other accounting period to process. The default is the current month.

Description

The **cmonacct** command performs monthly or periodic SP system accounting. The intervals are set in the **crontab** file. You can set the **cron** daemon to run the **cmonacct** command once each month or at some other specified time period. By default, if accounting is enabled for at least one node, **cmonacct** executes on the first day of every month.

The **cmonacct** command creates summary files under the **/var/adm/cacct/fiscal** directory and restarts summary files under the **/var/adm/cacct/sum** directory, the cumulative summary to which daily reports are appended.

cprdaily

Purpose

cprdaily – Creates an ASCII report of the previous day's accounting data.

Syntax

cprdaily [–**c**] [[–**l**] [*yyyymmdd*]]

Flags

–**c** Reports exceptional resource usage by command. This flag may be used only on the current day's accounting data.

–**l** Reports exceptional usage by login ID for the specified date specified in *mmdd* variable, if other than current day's reporting is desired. (This is lowercase **l**, as in **l**ist.)

Operands

yyyymmdd Specifies the date for exceptional usage report if other than the current date.

Description

This command is called by the **crunacct** command to format an ASCII report of the previous day's accounting data for all nodes. The report resides in the /**var**/**adm**/**cacct**/**sum**/**rprt***yyyymmdd* file, where *yyyymmdd* specifies the year, month, and day of the report.

create_krb_files

Purpose

create_krb_files – Creates the necessary **krb_srvtab** and **tftp** access files on the Network Installation Management (NIM) master.

Syntax

create_krb_files [–**h**]

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken.

Operands

None.

Description

Use this command on a boot/install server (including the control workstation). On the server, it creates the Kerberos **krb_srvtab** file for each boot/install client of that server and also updates the **/etc/tftpaccess.ctl** file on the server.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/create_krb_files

Related Information

Commands: **setup_server**

Examples

To create or update the **krb_srvtab** and **tftp** access files on a boot/install server, enter the following command on that server:

```
create_krb_files
```

createhsd

Purpose

createhsd – Creates one data striping device (HSD) that encompasses two or more IBM Virtual Shared Disks.

Syntax

createhsd –**n** {*node_list* | **ALL**} –**s** *size_in_MB* –**g** *vg_name*
–**t** *stripe_size_in_KB* [–**c** *vsds_per_node* [–**A**]] [–**m** *mirror_cnt*]
[–**d** *hsd_name*] [–**l** *lv_name_prefix*] [–**S**] [–**o cache** | **nocache**]
[–**T** *lp_size_in_MB*]

Flags

Note: Some examples shown in this list do not contain enough flags to be executable. They are shown in an incomplete form to illustrate specific flags.

–**n** Specifies the numbers of the nodes on which you are creating a data striping device (HSD). The backup node for the underlying IBM Virtual Shared Disks cannot be the same as the primary node.

node_list Given in the format **P**/**S:***hdisk_list1*+*hdisk_list2*/, where **P** is the primary node, **S**, if specified, is the backup (secondary) node, *hdisk_list1* is the list of local physical disks in the logical volume on the primary, and *hdisk_list1*+*hdisk_list2* is the list of local physical disks in the volume group on the primary, if you want to have more disks in the volume group than are needed for the logical volume.

The sequence in which nodes are listed determines the names given to the IBM Virtual Shared Disks; for example:

```
createhsd -n 1,6,4 -d DATA
```

(with the *hsd_prefix* DATA) creates IBM Virtual Shared Disks DATA1n1 on node 1, DATA2n6 on node 6, and DATA3n4 on node 4, which make up a single HSD named DATA. To create volume groups that span specified disks on nodes 1, 2, and 3 of a system with backup on nodes 4, 5, and 6 of the same system, and that make up a single HSD, enter:

```
createhsd -n 1/4:hdisk1,hdisk2,hdisk3/,2/5:hdisk5,hdisk6, \
hdisk7/,3/6:hdisk2,hdisk4,hdisk6/ -d DATA -s 12 -g OLD -t 4096
```

This command is shown on two lines here, but you must enter it without any spaces between the items in *node_list*.

The command creates:

- IBM Virtual Shared Disk DATA1n1b4 with logical volume lvDATA1n1b4 on a volume group with the local volume group name OLD on node 1, exported to node 4. The global volume group name is OLDn1b4.
- IBM Virtual Shared Disk DATA2n2b5 with logical volume lvDATA2n2b5 on a volume group with the local volume group name OLD on node 2, exported to node 5. The global volume group name is OLDn2b5.

- IBM Virtual Shared Disk DATA3n3b6 with logical volume lvDATA3n3b6 on a volume group with the local volume group name OLD on node 3, exported to node 6. The global volume group name is OLDn3b6.
- An HSD named DATA that comprises the three IBM Virtual Shared Disks listed previously. The HSD is 12MB in total usable size and has a 512KB stripe size.

If a volume group is already created and the combined physical hdisk lists contain disks that are not needed for the logical volume, those hdisks are added to the volume group. If the volume group has not already been created, **createhsd** creates a volume group that spans *hdisk_list1+hdisk_list2*.

Backup nodes cannot use the same physical disk as the primary does to serve IBM Virtual Shared Disks.

ALL Specifies that you are creating HSDs on all nodes in the system or system partition. If you use **ALL**, you can't assign backup nodes for the disks.

–s Specifies the total usable size of the HSD in MB. Unless **–S** is specified, **createhsd** adds at least a stripe size to each IBM Virtual Shared Disk's size for each HSD.

–g Specifies the Logical Volume Manager (LVM) volume group name, or local volume group name. This name is concatenated with the node number to form the global volume group name (VSD_GVG). For example:

```
createhsd -n 6 -g VSDVG
```

creates a volume group with the IBM Virtual Shared Disk global volume group name VSDVGn6 and the local AIX volume group name VSDVG. The node number is added to the prefix for the global volume group name to avoid name conflicts with the name used for this volume group on other nodes. If a backup node exists, the global volume group name will be concatenated with the backup node number as well as the primary. For example:

```
createhsd -n 6/3/ -g VSDVG
```

creates VSDVGn6b3, where the primary node is node 6 and the backup node for this volume group is node 3.

–c Specifies the number of IBM Virtual Shared Disks to be created on each node. If *number_of_vsds_per_node* is not specified, one IBM Virtual Shared Disk is created for each node specified on **createvsd**. If more than one IBM Virtual Shared Disk is to be created for each node, the names will be allocated cyclically. For example:

```
createhsd -n 1,6 -c 2 -d DATA
```

creates IBM Virtual Shared Disks DATA1n1 on node 1, DATA2n6 on node 6, DATA3n1 on node 1, and DATA4n6 on node 6 and uses them to make up the HSD DATA.

–A Specifies that IBM Virtual Shared Disk names will be allocated to each node in turn. For example:

```
createhsd -n 1,6 -c 2 -g DATA
```

creates DATA1n1 and DATA2n1 on node 1, and DATA3n6 and DATA4n6 on node 6.

–o Specifies the use of either the **cache** or the **nocache** option for the underlying IBM Virtual Shared Disks.

–m Specifies the LVM mirroring count. The mirroring count sets the number of physical partitions allocated to each logical partition. The range is from 1 to 3. If **–m** is not specified, the count is set to 1.

–t Specifies the stripe size in kilobytes that an HSD will use. The stripe size must be a multiple of 4KB and less than or equal to 1GB.

–d Specifies the name assigned to the created HSD. It is used as the IBM Virtual Shared Disk prefix name (the **–v** in **createvsd**). If an HSD name is not specified, a default name, *x*HsD is used, where x denotes a sequence number.

The command:

```
createhsd -n 1,2 -h DATA
```

creates two IBM Virtual Shared Disks, DATA1n1 and DATA2n2. These IBM Virtual Shared Disks make up one HSD named DATA.

–l Overrides the prefix *lvx* that is given by default to a logical volume by the **createvsd** command, where *x* is the IBM Virtual Shared Disk name prefix specified by *vsd_name_prefix* or the default (`vsd`). For example:

```
createhsd -n 1 -v DATA
```

creates one IBM Virtual Shared Disk on node 1 named DATA1n1 with an underlying logical volume lvDATA1n1. If the command

```
createhsd -n 1 -v DATA -l new
```

is used, the IBM Virtual Shared Disk on node 1 is still named DATA1n1, but the underlying logical volume is named lvnew1n1.

It is usually more helpful **not** to specify **–l**, so that your lists of IBM Virtual Shared Disk names and logical volume names are easy to associate with each other and you avoid naming conflicts.

–S Specifies that the HSD overrides the default skip option and does not skip the first stripe to protect the first LVM Control Block (LVCB).

–T Specifies the size of the physical partition in the Logical Volume Manager logical volume group and also the logical partition size (they will be the same) in megabytes. You must select a power of 2 in the range 2—256. The default is 4MB.

The Logical Volume Manager limits the number of physical partitions to 1016 per disk. If a disk is greater than 4 gigabytes in size, the physical partition size must be greater than 4MB to keep the number of partitions under the limit.

Operands

None.

Description

This command utilizes the sysctl facility.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit createhsd_dialog
```

or

```
smit vsd_data
```

and Select the Create an HSD option with the `vsd_data` fastpath.

Files

/usr/lpp/csd/bin/createhsd
Specifies the command file.

Standard Output

For the following command:

```
createhsd -n 1/:hdisk2,hdisk3/ -g twinVG -s 1600 -t 8 -S -l \
twinLV -d twinHSD -c 4
```

The messages returned to standard output are:

```
OK:0:vsdvg -g twinVGn1 twinVG 1

OK:0:defvsd twinLV1n1 twinVGn1 twinHSD1n1 cache
OK:0:defvsd twinLV2n1 twinVGn1 twinHSD2n1 cache
OK:0:defvsd twinLV3n1 twinVGn1 twinHSD3n1 cache
OK:0:defvsd twinLV4n1 twinVGn1 twinHSD4n1 cache

OK:createvsd { -n 1/:hdisk2,hdisk3/ -s 401 -T 4 -g twinVG
-c 4 -v twinHSD -l twinLV -o cache -K }

OK:0:defhsd twinHSD not_protect_lvcb 8192 twinHSD1n1 twinHSD2n1
twinHSD3n1 twinHSD4n1
```

Exit Values

0 Indicates the successful completion of the command.

–1 Indicates that an error occurred.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Restrictions

1. The backup node cannot be the same as the primary node.
2. The last character of *hsd_name* cannot be numeric.
3. The *vsd_name_prefix* cannot contain the character '**.**'. See the **createvsd -v** option for details.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **createvsd**, **defhsd**, **vsdvg**

Examples

To create six 4MB IBM Virtual Shared Disks and their underlying logical volumes with a prefix of TEMP, as well as an HSD comprising those IBM Virtual Shared Disks (24MB overall) with a stripe size of 32KB, enter the following (assuming that no previous IBM Virtual Shared Disks are defined with the TEMP prefix):

```
createhsd -n 3,4,7/8/ -c 2 -s 1024 -g vsdvg -d TEMP -t 32
```

This creates the following IBM Virtual Shared Disks:

- TEMP1n3 with a logical volume lvTEMP1n3 defined in Logical Volume Manager volume group vsdvg3 of node 3.
- TEMP2n4 with a logical volume lvTEMP2n4 defined in volume group vsdvg4 of node 4.
- TEMP3n7b8 with a logical volume lvTEMP3n7b8 defined in volume group vsdvg7b8 of node 7 and imported to node8
- TEMP4n3 with a logical volume lvTEMP4n3 defined in volume group vsdvg3 of node 3
- TEMP5n4 with a logical volume lvTEMP5n4 defined in volume group vsdvg4 of node 4
- TEMP6n7b8 IBM Virtual Shared Disk with a logical volume lvTEMP6n7b8 defined in volume group vsdvg7b8 of node 7 and imported to node8

and the HSD:

- TEMP with a stripe size of 32KB that includes the preceding IBM Virtual Shared Disks in the order listed.

Note: TEMP does not write to the first 32KB of each of its IBM Virtual Shared Disks.

createvsd

Purpose

createvsd – Creates a set of IBM Virtual Shared Disks, with their associated logical volumes, and puts information about them into the System Data Repository (SDR).

Syntax

Note: Some examples shown in this list do not contain enough flags to be executable. They are shown in an incomplete form to illustrate specific flags.

createvsd –**n** {*node_list* | **ALL**}
–**s** *size_in_MB* –**g** *volume_group_name*
[{–**c** *number_of_vsds_per_node* | –**L**}]
[–**o cache** | **nocache**] [–**m** *mirror_count*]
[–**p** *lvm_stripe_size*] [–**v** *vsd_name_prefix*]
[–**l** *lv_name_prefix*] [–**T** *lp_size_in_MB*]

Flags

–**n** Specifies the nodes on which you are creating IBM Virtual Shared Disks. The backup node cannot be the same as the primary node.

node_list Given in the format **P**/**S:***hdisk_list1*+*hdisk_list2*/, where **P** is the primary node, **S**, if specified, is the backup (secondary) node, *hdisk_list1* is the list of local physical disks in the logical volume on the primary, and *hdisk_list1*+*hdisk_list2* is the list of local physical disks in the volume group on the primary, if you want to have more disks in the volume group than are needed for the logical volume. The sequence in which nodes are listed determines the names given to the IBM Virtual Shared Disks. For example:

```
createvsd -n 1,6,4 -v PRE
```

(with the *vsd_prefix* PRE) creates IBM Virtual Shared Disks PRE1n1 on node 1, PRE2n6 on node 6, and PRE3n4 on node 4.

To create a volume group that spans hdisk2, hdisk3, and hdisk4 on node 1, with a backup on node 3, enter:

```
createvsd -n 1/3:hdisk2,hdisk3,hdisk4/ -v DATA
```

This command creates:

- IBM Virtual Shared Disk DATA1n1b3 with logical volume lvDATA1n1b3 on a volume group with the global volume group name DATA1n1b3 on node 1, exported to node 3. The Logical Volume Manager (LVM) volume group name is DATA. The logical volumes span hdisk2, hdisk3, and hdisk4.

To create volume groups just like that one on nodes 1, 2, and 3 of a system with backup on nodes 4, 5, and 6 of the same system, enter:

```
createvsd -n 1/4:hdisk1,hdisk2,hdisk3/,2/5:hdisk5,hdisk6, \
hdisk7/,3/6:hdisk2,hdisk4,hdisk6/ -v DATA
```

This command is shown on two lines here, but you must enter it without any spaces between the items in *node_list*.

The command creates:

- IBM Virtual Shared Disk DATA1n1b4 with logical volume lvDATA1n1b4 on a volume group with the local volume group name DATA on node 1, exported to node 4. The global volume group name is DATAn1b4.
- IBM Virtual Shared Disk DATA2n2b5 with logical volume lvDATA2n2b5 on a volume group with the local volume group name DATA on node 2, exported to node 5. The global volume group name is DATAn2b5.
- IBM Virtual Shared Disk DATA3n3b6 with logical volume lvDATA3n3b6 on a volume group with the local volume group name DATA on node 3, exported to node 6. The global volume group name is DATAn3b6.

To create an IBM Virtual Shared Disk where the logical volume spans only two of the physical disks in the volume group, enter:

```
createvsd -n 1/3:hdisk1,hdisk2+hdisk3/ -v DATA
```

This command creates the IBM Virtual Shared Disk DATA1n1b3 with logical volume lvDATA1n1b3 spanning hdisk1 and hdisk2 in the volume group DATA, which includes hdisk1, hdisk2, and hdisk3. It exports the volume group DATA to node 3.

If a volume group is already created and the combined physical hdisk lists contain disks that are not needed for the logical volume, those hdisks are added to the volume group. If the volume group has not already been created, **createvsd** creates a volume group that spans *hdisk_list1*+*hdisk_list2*.

Backup nodes cannot use the same physical disk as the primary does to serve IBM Virtual Shared Disks.

ALL Specifies that you are creating IBM Virtual Shared Disks on all nodes in the system or system partition. No backup nodes are assigned if you use this operand. The IBM Virtual Shared Disks will be created on all the physical disks attached to the nodes in *node_list* (you cannot specify which physical disks to use.)

–s Specifies the size in megabytes of each IBM Virtual Shared Disk

–g Specifies the Logical Volume Manager (LVM) volume group name. This name is concatenated with the node number to produce the global volume group name. For example:

```
createvsd -n 6 -g VSDVG
```

creates a volume group with the local volume group name VSDVG and the global volume group name VSDVG1n6 on node 6. The node number is added to the prefix to avoid name conflicts when a backup node takes over a volume group. If a backup node exists, the global volume group name will be concatenated with the backup node number as well as the primary. For example:

```
createvsd -n 6/3/ -g VSDVG
```

creates a volume group with the local volume group name VSDVG and the global volume group name VSDVGn6b3. The primary node is node 6 and the backup node for this volume group is node 3.

–c Specifies the number of IBM Virtual Shared Disks to be created on each node. If *number_of_vsds_per_node* is not specified, one IBM Virtual Shared Disk is created for each node specified on **createvsd**. If more than one IBM Virtual Shared Disk is to be created for each node, the names will be allocated alternately. For example:

```
createvsd -n 1,6 -c 2 -v DATA
```

creates IBM Virtual Shared Disks DATA1n1 on node 1, DATA2n6 on node 6, DATA3n1 on node 1, and DATA4n6 on node 6.

–L The **-L** option allows you to create one IBM Virtual Shared Disk on each node without using sequential numbers, for locally-accessed IBM Virtual Shared Disks.

–A Specifies that IBM Virtual Shared Disk names will be allocated to each node in turn, for example:

```
createvsd -n 1,6 -c 2 -h DATA
```

creates DATA1n1 and DATA2n1 on node 1, and DATA3n6 and DATA4n6 on node 6.

–o Specifies either the **cache** or the **nocache** option.

–m Specifies the LVM mirroring count. The mirroring count sets the number of physical partitions allocated to each logical partition. The range is from 1 to 3 and the default value is 1.

–p Specifies the LVM stripe size. If this flag is not specified, the logical volumes are not striped. In order to use striping, the node on which the IBM Virtual Shared Disks are defined must have more than one physical disk.

–v Specifies a prefix to be given to the names of the created IBM Virtual Shared Disks. This prefix will be concatenated with the IBM Virtual Shared Disk number, node number, and backup node number, if a backup disk is specified. For example, if the prefix PRE is given to an IBM Virtual Shared Disk created on node 1 and there are already two IBM Virtual Shared Disks with this prefix across the partition, the new IBM Virtual Shared Disk name will be PRE3n1. The name given to the underlying logical volume will be lvPRE3n1, unless the **–l** flag is used. The **createvsd** command continues to sequence IBM Virtual Shared Disk names from the last PRE-prefixed IBM Virtual Shared Disk

If **-v** is not specified, the prefix `vsd` is used.

Note: The last character of the *vsd_name_prefix* cannot be a digit. Otherwise, the 11th IBM Virtual Shared Disk with the prefix PRE would have the same name as the first IBM Virtual Shared Disk with the prefix PRE1. Nor can the *vsd_name_prefix* contain the character '**.**' because '**.**' can be any character in regular expressions.

–l Overrides the prefix *lvx* that is given by default to a logical volume by the **createvsd** command, where *x* is the IBM Virtual Shared Disk name prefix specified by *vsd_name_prefix* or the default (vsd). For example:

```
createvsd -n 1 -v DATA
```

creates one IBM Virtual Shared Disk on node 1 named DATA1n1 with an underlying logical volume lvDATA1n1. If the command

```
createvsd -n 1 -v DATA -l new
```

is used, the IBM Virtual Shared Disk on node 1 is still named DATA1n1, but the underlying logical volume is named lvnew1n1.

It is usually more helpful **not** to specify **-l**, so that your lists of IBM Virtual Shared Disk names and logical volume names are easy to associate with each other and you avoid naming conflicts.

–T Specifies the size of the physical partition in the Logical Volume Manager logical volume group and also the logical partition size (they will be the same) in megabytes. You must select a power of 2 in the range 2—256. The default is 4MB.

The Logical Volume Manager limits the number of physical partitions to 1016 per disk. If a disk is greater than 4 gigabytes in size, the physical partition size must be greater than 4MB to keep the number of partitions under the limit.

Operands

None.

Description

Use this command to create a volume group with the specified name (if one does not already exist) and creates a logical volume of size **s** within that volume group.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_data
```

and select the Create an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/createvsd Specifies the command file.

Standard Output

For the following command:

```
createvsd -n 1/:hdisk1/ -g testvg -s 16 -T 8 -l lvtest -v test -c 4
```

The messages returned to standard output are:

```
OK:0:vsdvg -g testvgn1 testvg 1
OK:0:defvsd lvtest1n1 testvgn1 test1n1 cache
OK:0:defvsd lvtest2n1 testvgn1 test2n1 cache
OK:0:defvsd lvtest3n1 testvgn1 test3n1 cache
OK:0:defvsd lvtest4n1 testvgn1 test4n1 cache
```

For the following command:

```
createvsd -n 1/:hdisk1/ -g testvg -s 16 -T 8 -l lvtest -v test -c 4
```

The messages returned to standard output are:

```
OK:0:defvsd lvtest5n1 testvgn1 test5n1 cache
OK:0:defvsd lvtest6n1 testvgn1 test6n1 cache
OK:0:defvsd lvtest7n1 testvgn1 test7n1 cache
OK:0:defvsd lvtest8n1 testvgn1 test8n1 cache
```

Exit Values

0 Indicates the successful completion of the command.

–1 Indicates that an error occurred.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Restrictions

1. The backup node cannot be the same as the primary node.
2. The last character of *vsd_name_prefix* cannot be numeric.
3. The *vsd_name_prefix* cannot contain the character '**.**'.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defvsd**, **vsdvg**

Examples

To create two 4MB IBM Virtual Shared Disks on each of three primary nodes, one of which has a backup, enter:

```
createvsd -n 3,4,7/8/ -c 2 -s 4 -g vsdvg -v TEMP
```

This command creates the following IBM Virtual Shared Disks:

- TEMP1n3, with logical volume lvTEMP1n3 on a volume group with the global volume group name vsdvgn3 on node 3
- TEMP2n4, with logical volume lvTEMP2n4 on a volume group with the global volume group name vsdvgn4 on node 4
- TEMP3n7b8, with logical volume lvTEMP3n7b8 on a volume group with the global volume group name vsdvgn7b8 on node 7, also imported to node 8
- TEMP4n3, with logical volume lvTEMP4n3 on a volume group with the global volume group name vsdvgn3 on node 3
- TEMP5n4, with logical volume lvTEMP5n4 on a volume group with the global volume group name vsdvgn4 on node 4
- TEMP6n7b8, with logical volume lvTEMP6n7b8 on a volume group with the global volume group name vsdvgn7b8 on node 7, also imported to node 8

To create three IBM Virtual Shared Disks, where the logical volume created on node 3 spans fewer disks than the volume group does, enter:

```
createvsd -n 3,4/:hdisk1,hdisk2+hdisk3/,7/8/ -s 4 -g datavg -v USER
```

This command creates:

- USER1n3, with logical volume lvUSER1n3 defined on a volume group with the global volume group name datavgn3 on node 3.
- USER2n4, with logical volume lvUSER2n4 defined on a volume group with the global volume group name datavgn4 on node 4. datavgn4 spans hdisk1, hdisk2, and hdisk3. lvUSER2n4 spans hdisk1 and hdisk2.
- USER3n7b8, with logical volume lvUSER3n7b8 defined on a volume group with the global volume group name datavgn7b8 on node 7, also imported to node 8.
- If no volume group was defined on nodes 3 and 7 before this **createvsd** command was issued, the volume groups datavgn3 and datavgn7b8 are created with one 4MB partition from a single physical disk.

crunacct

Purpose

crunacct – Runs on the **acct_master** node to produce daily summary accounting reports and to accumulate accounting data for the fiscal period using merged accounting data from each node.

Syntax

crunacct *yyyymmdd*
[**SETUP** | **DELNODEDATA** | **MERGETACCT** | **CMS** | **USEREXIT** | **CLEANUP**]

Flags

SETUP Copies the files produced by **nrunacct** on each node to the **acct_master** node. For each node named by the string *node*, these files:

/var/adm/acct/nite/lineuse*YYYYMMDD*
/var/adm/acct/nite/reboots*YYYYMMDD*
/var/adm/acct/nite/daytacct*YYYYMMDD*
/var/adm/acct/sum/daycms*YYYYMMDD*
/var/adm/acct/sum/loginlog*YYYYMMDD*

are copied to the **acct_master** node to the following files:
/var/adm/cacct/*node***/nite/lineuse***YYYYMMDD*
/var/adm/cacct/*node***/nite/reboots***YYYYMMDD*
/var/adm/cacct/*node***/nite/daytacct***YYYYMMDD*
/var/adm/cacct/*node***/sum/daycms***YYYYMMDD*
/var/adm/cacct/*node***/sum/loginlog***YYYYMMDD*

for all *YYYYMMDD* prior or equal to the *YYYYMMDD* being processed.

DELNODEDATA
Deletes files that have been copied to the **acct_master** node in the SETUP step, as well as the associated **/var/adm/acct/statefile***YYYYMMDD* files.

MERGETACCT
Produces a daily total accounting file and merges this daily file into the total accounting file for the fiscal period, for each accounting class. If there are no defined accounting classes, the output of this step represents data for the entire SP system.

CMS Produces a daily command summary file and merges this daily file into the total command summary file for the fiscal period, for each accounting class. If there are no defined accounting classes, the output of this step represents data for the entire SP system.

It also creates an SP system version of the **loginlog** file, in which each line consists of a date, a user login name and a list of node names. The date is the date of the last accounting cycle during which the user, indicated by the associated login name, had at least one connect session in the SP system. The associated list of node

names indicates the nodes on which the user had a login session during that accounting cycle.

USEREXIT If the **/var/adm/csiteacct** shell file exists, calls it to perform site specific accounting procedures that are applicable to the **acct_master** node.

CLEANUP Prints a daily report of accounting activity and removes files that are no longer needed.

Operands

yyyymmdd Specifies the date for accounting to be rerun.

Description

In order for SP accounting to succeed each day, the **nrunacct** command must complete successfully on each node for which accounting is enabled and then the **crunacct** command must complete successfully on the **acct_master** node. However, this may not always be true. In particular, the following scenarios must be taken into account:

1. The **nrunacct** command does not complete successfully on some nodes for the current accounting cycle. This can be the result of an error during the execution of **nrunacct**, **nrunacct** not being executed at the proper time by **cron** or the node being down when **nrunacct** was scheduled to run.
2. The **acct_master** node is down or the **crunacct** command cannot be executed.

From the point of view of the **crunacct** command, the first scenario results in no accounting data being available from a node. The second scenario results in more than one day's accounting data being available from a node. If it is the case that no accounting data is available from a node, the policy of **crunacct** is that the error condition is reported and processing continues with data from the other nodes. If data cannot be obtained from at least X percent of nodes, then processing is terminated. "X" is referred to as the **spacct_actnode_thresh** attribute and can be set via a SMIT panel.

If node data for accounting cycle N is not available when **crunacct** executes and then becomes available to **crunacct** during accounting cycle N+1, the node data for both the N and N+1 accounting cycles is merged by **crunacct**. In general, **crunacct** merges all data from a node that has not yet been reported into the current accounting cycle, except as in the following case.

If it is the case that **crunacct** has not run for more than one accounting cycle, such that there are several day's data on each node, then the policy of **crunacct** is that it processes each accounting cycle's data to produce the normal output for each accounting cycle. For example, if **crunacct** has not executed for accounting cycles N and N+1, and it is now accounting cycle N+2, then **crunacct** first executes for accounting cycle N, then executes for accounting cycle N+1 and finally executes for accounting cycle N+2.

However, if the several accounting cycles span from the previous fiscal period to the current fiscal period, then only the accounting cycles that are part of the previous fiscal period are processed. The accounting cycles that are part of the current fiscal period are processed during the next night's execution of **crunacct**. Appropriate messages are provided in the **/var/adm/cacct/active** file so that the

administrator can execute **cmonacct** prior to the next night's execution of **crunacct**.

Restart Procedure

To restart the **crunacct** command after a failure, first check the **/var/adm/cacct/active***YYYYMMDD* file for diagnostic messages, and take appropriate actions. For example, if the log indicates that data was unavailable from a majority of nodes, and their corresponding **nrunacct** state file indicate a state other than complete, check their **/var/adm/acct/nite/active***YYYYMMDD* files for diagnostic messages and then fix any damaged data files, such as **pacct** or **wtmp**.

Remove the lock files and lastdate file (all in the **/var/adm/cacct** directory), before restarting the **crunacct** command. You must specify the **–r** and *YYYYMMDD* parameters if you are restarting the **crunacct** command. It specifies the date for which the **crunacct** command is to rerun accounting. The **crunacct** procedure determines the entry point for processing by reading the **/var/adm/cacct/statefile***YYYYMMDD* file. To override this default action, specify the desired state on the **crunacct** command line.

Files

/var/adm/cacct/activeYYYYMMDD
The **crunacct** message file.

/var/adm/cacct/fiscal_periods
Customer-defined file indicating start date of each fiscal period.

/var/adm/cacct/lastcycle
Contains last successful **crunacct** completed cycle.

/var/adm/cacct/lock*
Prevents simultaneous invocation of **crunacct**.

/var/adm/cacct/lastdate
Contains last date **crunacct** was run.

/var/adm/cacct/nite/statefileYYYYMMDD
Contains current state to process.

Security

Access Control: This command should grant execute (x) access only to members of the adm group.

Prerequisite Information

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see *IBM Parallel System Support Programs for AIX: Administration Guide*.

Related Information

Commands: **acctcms**, **acctcom**, **acctcon1**, **acctcon2**, **acctmerg**, **acctprc1**, **acctprc2**, **accton**, **crontab**, **fwtmp**, **nrunacct**

Daemon: **cron**

The System Accounting information found in *AIX Version 4.1 System Management Guide*.

Examples

1. To restart the SP system accounting procedures for a specific date, enter a command similar to the following:

   ```
   nohup /usr/lpp/ssp/bin/crunacct -r 19940601 2>> \
   /var/adm/cacct/nite/accterr &
   ```

   This example restarts **crunacct** for the day of June 1 (0601), 1994. The **crunacct** command reads the file **/var/adm/cacct/statefile19940601** to find out the state with which to begin. The **crunacct** command runs in the background (&), ignoring all INTERRUPT and QUIT signals (**nohup**). Standard error output (2) is added to the end (>>) of the **/var/adm/cacct/nite/accterr** file.

2. To restart the SP system accounting procedures for a particular date at a specific state, enter a command similar to the following:

   ```
   nohup /usr/lpp/ssp/bin/crunacct 19940601 CMS 2>> \
   /var/adm/cacct/nite/accterr &
   ```

   This example restarts the **crunacct** command for the day of June 1 (0601), 1994, starting with the CMS state. The **crunacct** command runs in the background (&), ignoring all INTERRUPT and QUIT signals (the **nohup** command). Standard error output (2) is added to the end (>>) of the **/var/adm/cacct/nite/accterr** file.

cshutdown

Purpose

cshutdown – Specifies the SP system **Shutdown** command.

Syntax

cshutdown [–**G**] [–**E**] [–**N** | –**R** | –**g**] [–**W** *seconds*] [–**X**] [–**Y**]
[–**F**] [–**h** | –**k** | –**m** | –**r** [–**C** *cstartup_options*]]
[–**T** *time* [–**M** *message_string*]]
target_nodes

Flags

–**C** *cstartup_options*
: Tells **cshutdown** to pass the *cstartup_options* to **cstartup** when the **cstartup** command is invoked after the *target_nodes* are halted. This flag is valid only when the –**r** (reboot) option is also specified. Any blanks in *cstartup_options* must be escaped or quoted.

–**E**
: Terminates processing if any nodes are found that are powered on, but not running (host_responds in the System Data Repository (SDR) shows a value of 0—node shows red on the system monitor). This includes nodes that may have been placed in maintenance (single-user) mode. Refer to the “Description” section for additional information.

  If you specify –**E**, you cannot specify –**X**.

–**G**
: Allows the specification of nodes to include one or more nodes outside the current system partition. If **ALL** is specified with –**G**, all nodes in the SP are shut down. If **ALL** is specified without –**G**, all nodes in the current system partition are shut down. If –**G** is specified with a list of nodes, all listed nodes are shut down regardless of the system partition in which they reside (subject to the restrictions of the sequence file). If –**G** is not specified and some of the specified target nodes are outside of the current system partition or some of the specified target nodes depend on nodes outside of the current system partition, none of the specified nodes are shut down.

–**g**
: Indicates that the *target_nodes* are specified as a named node group. If –**G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

–**F**
: Tells the **cshutdown** command to start the shut down immediately, without issuing warning messages to users.

–**h**
: Halts the target nodes. This is the default, unless overridden by the –**k**, –**m**, or –**r** flags.

–**k**
: Verifies the shutdown sequence file without shutting any node down. Special subsystems are not affected. There is no effect on a nonrunning target node. You can use **cshutdown** –**kF ALL** to test your /**etc**/**cshutSeq** file without actually shutting down any nodes and without sending messages to users.

–m Handles the request similar to a halt except that the last step, after syncing and unmounting file systems, is to bring the node to single user mode. There is no effect on a nonrunning target node.

–N Indicates that the *target_nodes* are specified as node numbers, not en0 host names. The node numbers can be specified as ranges, for example, 3-7 indicates nodes 3, 4, 5, 6, and 7.

–r Handles the request as a reboot. It performs the same operations as **–h**. Then it restarts the target nodes with **cstartup**. It does not power on a target node that was powered off at the time the **cshutdown** command was issued (it differs from the **cstartup** command, which powers on all specified nodes).

–R Indicates that *target_nodes* is a file that contains host identifiers. If you also use the **–N** flag, the file contains node numbers; otherwise, the file contains node names, specified as en0 host names.

[**–T** *time* [**–M** *message_string*]]
The **–T** flag specifies a *time* to start **cshutdown**, either as a number of minutes from now (**–T** *minutes*) or at the time in 24–hour format (**–T** *hh***:***mm*). If the **–T** flag is specified, then you can use **–T** *message_string* to specify a message for users on the target nodes. Any blanks in *message_string* must be escaped or quoted.

–W *seconds* Provides a time-out value for shutting down a leading node. In normal processing, **cshutdown** waits for a leading node to be completely halted before starting to shut down trailing nodes. If one or more leading nodes fails to shut down, the **cshutdown** command waits indefinitely. The **–W** flag tells **cshutdown** to wait only the specified number of *seconds* after starting to halt a leading node; after that time, **cshutdown** starts the halt process for the trailing nodes.

Notes:

1. Be careful to use time-out values large enough to allow a node to complete shutdown processing. Your time-out value should be at least several minutes long; shorter values may be transparently modified to a higher value.
2. If shutdown processing for a node does not complete within the time-out limit and **cshutdown** halts trailing nodes, the system may not function correctly.

If there are special subsystems, the same waiting procedure applies to subsystem sequencing in the subsystem phase.

–X Tells **cshutdown** that the state of nontarget nodes should not affect the result of the command. Use the **–X** flag to force **cshutdown** to shut down the target nodes if nontarget nodes listed in **/etc/cshutSeq** are gating the shutdown.

Note: If some critical nodes, but not the entire system, are forced to halt or reboot, the system may not function correctly.

–Y Tells **cshutdown** to ignore any failure codes from the special subsystem interfaces. Without this flag, if a special subsystem interface exits with a failure code, you receive a prompt allowing you to continue the operation, to quit, or to enter a subshell to investigate the failure. On return from the subshell, you are prompted with the same choices.

Operands

target_nodes Designates the target nodes to be operated on. It is the operand of the command, and must be the last token on the command line. In the absence of the **–R**, **–N**, or **–g** flags, *target_nodes* are specified as host names on the en0 Ethernet. Use **ALL** to designate the entire system. You must identify one or more *target_nodes*.

Description

Use this command to halt or reboot the entire system or any number of nodes in the system. The SP **cshutdown** command is analogous to the workstation **shutdown** command. Refer to the **shutdown** man page for a description of the **shutdown** command. The **cshutdown** command always powers off the nodes except while in Maintenance mode.

Note: If you bring a node down to maintenance mode, you must ensure file system integrity before rebooting the node.

The **cshutdown** command, which runs from the control workstation, cannot **rsh** to the node to perform the node shutdown phase processing. This includes the synchronization of the file systems. Therefore, you should issue the **sync** command three times in succession from the node console before running the **cshutdown** command. This is especially important if any files were created while the node was in maintenance mode.

To determine which nodes may be affected, issue the **spmon –d** command and look for a combination of power **on** and host_responds **no**.

The **cshutdown** command has these advantages over using the **shutdown** command to shut down each node of an SP:

- **cshutdown** provides a single point of control.

  Using one **cshutdown** command on the control workstation, you can shut down all or selected nodes.

- The sequencing of node shutdown and reboot is automated.

  You can use the /**etc**/**cshutSeq** file to control the order in which nodes are shut down, or you can let the system determine the order based on System Data Repository information about /**usr** servers and clients.

- Special subsystems can be notified of impending node shutdown.

  The /**etc**/**subsysSeq** file lists these special subsystems and describes any sequencing relationships between them.

Shutdown processing has these phases:

1. Notifying all users of the impending shutdown, then terminating all nonroot processes on the target nodes. Nonroot processes are sent a SIGTERM followed, 30 seconds later, by a SIGKILL. This gives user processes that handle SIGTERM a chance to do whatever cleanup is necessary.
2. Invoking any special subsystems, so they can perform any necessary shutdown activities. This phase follows the sequencing rules in **/etc/subsysSeq**. See *IBM Parallel System Support Programs for AIX: Administration Guide* for the format of the **/etc/subsysSeq** file.
3. Starting node phase shutdown. The node phase includes syncing and unmounting file systems and halting the nodes, following the sequencing rules in **/etc/cshutSeq**. See *IBM Parallel System Support Programs for AIX: Administration Guide* for the format of the **/etc/cshutSeq** file.
4. Rebooting the system, if requested by **–r** flag.

Files

The following files reside on the control workstation:

/usr/lpp/ssp/bin/cshutdown
The **cshutdown** command.

/etc/cshutSeq Describes the sequence in which the nodes should be shut down. Nodes not listed in the file are shut down concurrently with listed nodes. If the file is empty, all nodes are shut down concurrently. If the file does not exist, **cshutdown** uses the output of **seqfile** as a temporary sequencing default.

/etc/subsysSeq
Describes groups of special subsystems that need to be invoked in the subsystem phase of **cshutdown**. Also shows the sequence of invocation. Subsystems are represented by their invocation commands. If this file does not exist or is empty, no subsystem invocation is performed.

/var/adm/SPlogs/cs/cshut.*MMDDhhmmss***.***pid*
Road map of **cshutdown** command progress.

Restrictions

The **cshutdown** command can only be issued by root on the control workstation. The root user must issue the **kinit** command, specifying a principal name for which there is an entry in the **hardmon** ACLs file with control authorization for the frames to shut down. The **hardmon** and System Data Repository (SDR) must be running.

Results

The **cshutdown** command may be gated by the failure of some subsystems or nodes to complete shutdown. In this case, look in the file created:
/var/adm/SPlogs/cs/cshut.*MMDDhhmmss***.***pid*

MMDDhhmmss Time stamp.

pid The process ID of the **cshutdown** command.

If a file with the same name already exists (from a previous year), the **cshutdown** command overwrites the existing file.

Related Information

Commands: **cstartup**, **init**, **seqfile**, **shutdown**

Examples

1. For these examples, assume that **/etc/cshutSeq** contains the following lines:

   Group1 > Group2 > Group3
   Group1: A
   Group2: B
   Group3: C

   This defines 3 groups, Group1 through Group3, each containing a single node. The nodes names are A, B, and C. The sequence line `Group1 > Group2 > Group3` means that Group3 (node C) is shut down first. When Group3 is down, Group2 (node B) is shut down. When Group2 is down, then Group1 (node A) is shut down.

   Table 1 shows that the result of a **cshutdown** command depends on the flags specified on the command line, the initial state of each node, and the sequencing rules in **/etc/cshutSeq**. The shorthand notation A^{up} indicates that node A is up and running; A_{dn}indicates that node A is down.

Table 1. Examples of the cshutdown Command. The subscript up *means the node is powered up and running; the subscript* dn *means the node is not running.*

Initial State	Command Issued	Final State	Explanation
A^{up} B^{up} C^{up}	**cshutdown** A B C	A_{dn} B_{dn} C_{dn}	The command succeeds; the nodes are all down.
A^{up} B^{up} C_{dn}	**cshutdown** B	A^{up} B_{dn} C_{dn}	The command succeeds because C is already not running.
A^{up} B^{up} C_{dn}	**cshutdown** A	Unchanged	The command fails because B is still running.
A^{up} B^{up} C_{dn}	**cshutdown** –**X** A	A_{dn} B^{up} C_{dn}	The command succeeds because –**X** considers the sequencing of *only* the target nodes.

2. To shut down all the nodes in the SP system regardless of system partitions and the sequence file, enter:

   ```
   cshutdown -G ALL
   ```

3. To shut down nodes **1**, **9**, and **16—20** regardless of system partitions and subject to the restrictions of the sequence file, enter:

   ```
   cshutdown -G -N 1 9 16-20
   ```

   The command may fail if any node in the list depends on any node that is not on the list and that node is not shutdown.

4. To shut down all the nodes in the current system partition, enter:

   ```
   cshutdown ALL
   ```

   The command may fail if any node in the current system partition depends on nodes outside of the current system partition.

5. To shut down nodes **1**, **5**, and **6** in the current system partition, enter:

   ```
   cshutdown -N 1 5 6
   ```

The command may fail if any node in the list is not in the current system partition or depends on nodes outside of the current system partition.

6. Specify the –**X** flag to ignore the sequence file and force nodes **1**, **5**, and **6** to be shut down. The following command is successful even if node 5 is gated by a node that is not shut down or is outside the current system partition:

   ```
   cshutdown -X -N 1 5 6
   ```

7. To do a fast shut down on node 5 without sending a warning message to the user, enter:

   ```
   cshutdown -F -N 5
   ```

8. To verify the sequence file without shutting down any node, enter the –**k** flag as follows. If both the –**k** and –**F** flags are specified, the sequence file can be tested without actually shutting down any nodes and without issuing a warning message to the user.

   ```
   cshutdown -kF ALL
   ```

9. Specify the –**r** flag to halt the target nodes and restart them with **cstartup**. If necessary, specify the –**C** flag to provide *cstartup_options*. For example, to halt and restart nodes 12—16 with a time-out value of 300 seconds for the purpose of starting a leading node, enter:

   ```
   cshutdown -rN -C"-W 300" 12-16
   ```

10. To reboot all the nodes in the partition node group sleepy_nodes, enter:

    ```
    cshutdown -rg sleepy_nodes
    ```

CSS_test

Purpose

CSS_test – Verifies that the installation and configuration of the Communications Subsystem of the SP system completed successfully.

Syntax

CSS_test

Flags

None.

Operands

None.

Description

Use this command to verify that the Communications Subsystem component **ssp.css** of the SP system was correctly installed. **CSS_test** runs on the system partition set in SP_NAME.

A return code of 0 indicates that the test completed without a failure, but unexpected results may be noted on standard output and in the companion log file **/var/adm/SPlogs/CSS_test.log**. A return code of 1 indicates that a failure occurred.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

Files

/usr/lpp/ssp/bin/CSS_test
Path name of this command

/var/adm/SPlogs/CSS_test.log
Default log file

Related Information

Commands: **jm_install_verify**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_ctest**, **spmon_itest**

Examples

To verify the Communication Subsystem following installation, enter:

```
CSS_test
```

cstartup

Purpose

cstartup – Specifies the SP system **Startup** command.

Syntax

cstartup [–**E**] [–**G**] [–**k**] [–**N** | –**R** | –**g**] [–**S**] [–**W** *seconds*] [–**X**] [–**Z**] [–**z**] {*target_nodes* | [**ALL**]}

Flags

–**E** Starts up all nodes concurrently. Ignores the /**etc**/**cstartSeq** file, if one exists.

–**G** Allows the specification of nodes to include one or more nodes outside of the current system partition. If **ALL** is specified with –**G**, all nodes in the SP start up. If **ALL** is specified without –**G**, all nodes in the current system partition start up. If –**G** is specified with a list of nodes, all listed nodes start up regardless of the system partition in which they reside (subject to the restrictions of the sequence file). If –**G** is not specified and some of the specified target nodes are outside of the current system partition or some of the specified target nodes depend on nodes outside of the current system partition, none of the specified nodes are started up.

–**g** Indicates that the *target_nodes* are specified as a named node group. If –**G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

–**k** Checks the sequence data file; does not start up any nodes. If circular sequencing is detected, **cstartup** issues warning messages. You can use **cstartup** –**k ALL** to test your /**etc**/**cstartSeq** file without starting or resetting any nodes.

–**N** Indicates that the *target_nodes* are specified as node numbers, not en0 host names. The node numbers can be specified as ranges; for example, 3–7 is interpreted as nodes 3, 4, 5, 6, and 7.

–**R** Indicates that *target_nodes* is a file that contains the node identifiers.

–**S** Tells **cstartup** to ignore existing sequencing violations; some trailing *target_nodes* are already up and running. The *target_nodes* that are already up are left alone. The other *target_nodes* are started in sequence. This operation may cause the nodes involved to not interface properly with their dependent nodes. If you omit the –**S** flag and any *target_node* is already running before its leading node, **cstartup** fails without modifying the state of the system.

–**W** *seconds* Provides a timeout value for starting up a leading node. In normal processing, **cstartup** waits for a leading node to be completely started before initiating the startup of trailing nodes. If one or more *target_nodes* fail to come up, **cstartup** waits indefinitely. The –**W** flag tells **cstartup** to wait the specified amount of time after initiating the startup of a node; the command continues to start

other nodes, preserving the sequence in /**etc**/**cstartSeq**. The value you specify as *seconds* is added to a 3 minute (180 second) default wait period. Your value is a minimum; internal processing may cause the actual wait time to be slightly longer.

Note: Your system may still be usable if one or more nodes fails to complete startup, because the sequencing rules are preserved.

–X Starts up only the nodes listed on the command line even if there are nontarget nodes gating the system startup. If you do not specify the **–X** flag and there are sequence violations involving nontarget nodes, **cstartup** fails without modifying the state of the system.

Note: If some nodes but not the entire system are forced to start up this way, they may not function properly because of possible resource problems.

–Z If a *target_node* is already running at the time the **cstartup** command is issued, this flag tells **cstartup** to reset the node. This operation is disruptive to any processes running on the node. If you omit the **–Z** flag and any *target_node* is already running, **cstartup** fails without modifying the state of the system.

–z If a *target_node* is already running at the time the **cstartup** command is issued, this flag tells **cstartup** to reset the node if the node is dependent on a node that is down when **cstartup** is issued, but leave the node alone if the node is to be started up ahead of any down node. This operation is disruptive to any processes running on the node being reset. This operation correctly resets the node-startup sequencing with minimum disruption to the system. If you omit the **–z** flag and any *target_node* is already running, **cstartup** fails without modifying the state of the system.

Operands

target_nodes Designates the target nodes to be operated on. It is the operand of the command, and must be the last token on the command line. In the absence of the **–R**, **–N**, or **–g** flags, *target_nodes* are specified as host names on the en0 Ethernet. The string **ALL** can be used to designate all nodes in the SP system. You must identify one or more *target_nodes*.

Description

Caution!

The **cstartup** command attempts to power on nodes that are powered off. This has safety implications if someone is working on the nodes. Proper precautions should be taken when using this command.

The **cstartup** command starts up the entire system or any number of nodes in the system. If a node is not powered on, startup means powering on the node. If the node is already powered on and not running, startup means resetting the node.

The **/etc/cstartSeq** file specifies the sequence in which the nodes are started up. See *IBM Parallel System Support Programs for AIX: Administration Guide* for the format of the **/etc/cstartSeq** file.

You can use the **–SXZ** flags to violate the **cstartup** sequence intentionally. See Table 2 on page 52 for examples of the effect of these flags.

Files

The following files reside on the control workstation:

/usr/lpp/ssp/bin/cstartup
The **cstartup** command.

/etc/cstartSeq Describes the sequence in which the nodes should be started. Nodes not listed in the file are started up concurrently with listed nodes. If the file is empty, all nodes are started up concurrently. If the file does not exist, **cstartup** uses the output of **seqfile** as a temporary sequencing default.

/var/adm/SPlogs/cs/cstart.*MMDDhhmmss***.***pid*
Road map of **cstartup** command progress.

Restrictions

The **cstartup** command can only be issued by root on the control workstation. The root user must issue the **kinit** command, specifying a principal name for which there is an entry in the **hardmon** ACLs file with control authorization for the frames to start up. The **hardmon** and System Data Repository (SDR) must be running.

Results

The **/var/adm/SPlogs/cs/cstart.***MMDDhhmmss***.***pid* file contains the results of **cstartup**.

MMDDhhmmss The time stamp.

pid The process ID of the **cstartup** command.

If the command fails, examine this file to see which steps were completed. If a file with the same name already exists (from a previous year), the **cstartup** command overwrites the existing file.

Related Information

Commands: **cshutdown**, **init**, **seqfile**

Examples

1. For these examples, assume that **/etc/cstartSeq** specifies the following startup sequence:

   Group1 > Group2 > Group3 > Group4 > Group5
   Group1: A
   Group2: B
   Group3: C
   Group4: D
   Group5: E

   This defines five groups, Group1 through Group5, each containing a single node. The nodes names are A, B, C, D, and E. The sequence line

`Group1 > Group2 > Group3 > Group4 > Group5` means that Group1 (node A) is started first. When Group1 is up, Group2 (node B) is started. When Group2 is up, then Group3 (node C) is started, and so on.

Table 2 shows that the result of a **cstartup** command depends on the flags specified on the command line, the initial state of each node, and the sequencing rules in **/etc/cstartSeq**. The shorthand notation A^{up} indicates that A is powered up and running; A_{dn}indicates that A is not running.

Table 2 (Page 1 of 2). Examples of the cstartup Command. The subscript up *means the node is up; the subscript* dn *means the node is down.*

Initial State	Command Issued	Final State	Explanation
A_{dn} B_{dn} C_{dn} D_{dn} E_{dn}	**cstartup** A B C D E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds; the nodes are all up.
A^{up} B^{up} C_{dn} D_{dn} E_{dn}	**cstartup** A B C D E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds, C, D, and E are started up.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** A B C D E	Unchanged	The command fails because D was already up before C.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** –**S** A B C D E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds because –**S** ignores sequencing violations.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** –**Z** A B C D E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds because –**Z** resets running nodes.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** C E	Unchanged	The command fails because node D was already up before node C.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** –**S** C E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds because –**S** ignores sequencing violations.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** –**X** C E	A^{up} B^{up} C^{up} D^{up} E^{up}	The command succeeds because –**X** considers the sequencing of *only* the target nodes.
A^{up} B^{up} C_{dn} D^{up} E_{dn}	**cstartup** –**Z** C E	unchanged	The command fails because resetting C or E does not correct the sequence violation.
A^{up} B^{up} C_{dn} D_{dn} E_{dn}	**cstartup** C E	unchanged	The command fails because D is gating E. Node C is not started either.
A^{up} B^{up} C_{dn} D_{dn} E_{dn}	**cstartup** –**S** C E	unchanged	The command fails because D is gating E. Node C is not started either.
A^{up} B^{up} C_{dn} D_{dn} E_{dn}	**cstartup** –**X** C E	A^{up} B^{up} C^{up} D_{dn} E^{up}	The command succeeds and starts up only the explicit targets, C and E.

Table 2 (Page 2 of 2). Examples of the cstartup Command. The subscript up *means the node is up; the subscript* dn *means the node is down.*

Initial State	Command Issued	Final State	Explanation
A_{up} B_{up} C_{dn} D_{dn} E_{dn}	**cstartup** –**Z** C E	unchanged	The command fails because D is gating E. Node C is not started either.

2. To start up all the nodes in the SP system regardless of system partitions and the sequence file, enter:

   ```
   cstartup -G ALL
   ```

3. To start up nodes **1**, **9**, and **16—20** regardless of system partitions and subject to the restrictions of the sequence file, enter:

   ```
   cstartup -G -N 1 9 16-20
   ```

   The command may fail if any node in the list depends on any node that is not on the list and that node is not started up.

4. To start up all the nodes in the current system partition, enter:

   ```
   cstartup ALL
   ```

   The command may fail if any node in the current system partition depends on nodes outside of the current system partition.

5. To start up nodes **1**, **5**, and **6** in the current system partition, enter:

   ```
   cstartup -N 1 5 6
   ```

   The command may fail if any node in the list is not in the current system partition or depends on nodes outside of the current system partition.

6. Specify the –**X** flag to ignore the sequence file and force nodes **1**, **5**, and **6** to be started up. The following command is successful even if node 5 is gated by a node that is not started up or is outside the current system partition:

   ```
   cstartup -X -N 1 5 6
   ```

7. To verify the sequence file without actually starting up or resetting any nodes, enter the –**k** flag as follows:

   ```
   cstartup -k ALL
   ```

8. To ignore the sequence file and start up all the target nodes concurrently, use the –**E** flag. For example, to start up all the nodes in the current system partition concurrently, enter:

   ```
   cstartup -E ALL
   ```

9. To start up all nodes in the system node group sleepy_nodes, enter:

   ```
   cstartup -Gg sleepy_nodes
   ```

ctlhsd

Purpose

ctlhsd – Sets the data striping device (HSD) for the IBM Virtual Shared Disks operational parameters or reset statistics.

Syntax

ctlhsd [–**p** *parallel_level* | –**v** *hsd_name* ... | –**C** | –**V**]

Flags

no option Displays the current parallelism level, the number of reworked requests, and the number of requests that are not at a page boundary.

–**p** *parallel_level*
Sets the HSD device driver's parallelism level as the specified value of the *parallel_level*.

–**v** *hsd_name* **...**
Resets the statistics in the number of reads and writes on the specified HSDs.

–**C** Resets the HSD device drivers counters in the number of reworked requests and the number of read/write requests that are not at a page boundary.

–**V** Resets all the configured HSD's statistics in the number of read and write requests.

Operands

None.

Description

Use this command to set the parallelism level and to reset the statistics of the data striping device (HSD) for the IBM Virtual Shared Disk. When specified with no arguments, it displays the the current parallelism level, the number of reworked requests, and the number of requests that were not at a page boundary. When **ctlhsd** is used to reset the statistics of the device driver, or a particular device, or all the configured data striping devices on the system, it will not suspend all the underlying IBM Virtual Shared Disks. In other words, the user should make sure that there are no I/O activities on the IBM Virtual Shared Disks.

Use **lshsd** –**s** to display the statistics on the number of read and write requests at the underlying IBM Virtual Shared Disks in an HSD or all HSDs. Use the –**v** or –**V** flag to reset these counters.

Files

/usr/lpp/csd/bin/ctlhsd
Specifies the command file.

Security

You must have root privilege to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsd**, **lshsd**, **lsvsd**, **resumevsd**, **suspendvsd**, **ucfghsd**

Examples

To display the current parallelism level and counter, enter:

```
ctlhsd
```

The system displays a message similar to the following:

```
The current parallelism level is 9.
The number of READ requests not at page boundary is 0.
The number of WRITE requests not at page boundary is 0.
```

ctlvsd

Purpose

ctlvsd – Sets IBM Virtual Shared Disk operational parameters.

Syntax

ctlvsd [–**t**] | [–**T**] | [–**c** *NewCacheSize* | –**r** *node_number*... | –**R** | –**k** *node_number*... –**K** | –**p** *1—9* | –**M** *max_IP_msg_size* | –**v** *vsd_name* ... | –**C** | –**V**]

Flags

–**t** Lists the current routing table and mbuf headers cached by the IBM Virtual Shared Disk driver.

–**T** Clears or releases all cached routes.

–**c** Sets the cache size to the new value. Only increasing the cache size up to the maximum value is supported. The initial value of the cache size is the *init_cache_buffer_count* from the SDR Node object for the node.

–**r** Resets the outgoing and expected sequence numbers for the nodes specified on the node on which the command is run. Use this flag when another node has either been rebooted, cast out or all IBM Virtual Shared Disks have been reconfigured on that node. The specified nodes are also cast in.

–**R** Resets the outgoing and expected sequence number for all nodes on the node on which the command is run. Use this flag after rebooting the node. All nodes in the IBM Virtual Shared Disk network will be cast in.

–**k** Casts out the node numbers specified on the local node. The local node ignores requests from cast out nodes. Use –**r** to cast nodes back in.

Note: Before using this flag, refer to the “Restrictions” section that follows.

–**K** Casts out all nodes on the local node. Local requests are still honored.

Note: Before using this flag, refer to the “Restrictions” section that follows.

–**p** Sets the level of IBM Virtual Shared Disk parallelism to the number specified. The valid range is 1 to 9. The default is 9. A larger value can potentially give better response time to large requests. (Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for more information regarding tuning IBM Virtual Shared Disk performance.)

This value is the *buf_cnt* parameter on the **uphysio** call the IBM Virtual Shared Disk IP device driver makes in the kernel. Use **statvsd** to display the current value on the node on which the command is run.

−M Sets the IBM Virtual Shared Disk *max_IP_msg_size*. This is the largest sized block of data the IBM Virtual Shared Disk sends over the network for an I/O request. This limit also affects local IBM Virtual Shared Disk I/O block size. The value must be a multiple of 512 and between 512 and 65024 (64KB-512KB). For the SP High Performance Switch, 65024 is recommended, and 24576 (24KB) for token-ring or Ethernet networks. (Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for more information regarding tuning IBM Virtual Shared Disk performance.) Use **statvsd** to display the current value on the node on which the command is run. Set to the same value on all nodes.

−v *vsd_name* **...**
Resets the statistics in the number of read and write requests on the specified IBM Virtual Shared Disks.

−C Resets the IBM Virtual Shared Disk device driver counters displayed by the **statvsd** command. Exceptions are the outgoing and expected request sequence numbers among the client and server nodes.

−V Resets all the configured IBM Virtual Shared Disk's statistics in the number of read and write requests.

Operands

None.

Description

The **ctlvsd** command changes some parameters of the IBM Virtual Shared Disk. When called with no arguments it displays the current and maximum cache buffer count, the request block count, the pbuf count, the minimum buddy buffer size, the maximum buddy buffer size as well as the overall size of the buddy buffer.

Use **statvsd** to display outgoing and expected sequence numbers and out cast status of other nodes as viewed by the node on which the command is run. It is best to **suspendvsd −a** on all nodes whose sequence numbers are being reset prior to actually resetting the sequence numbers. Be sure to use **resumevsd** on all IBM Virtual Shared Disks that were suspended after resetting the sequence numbers.

Initially, all sequence numbers are set to 0 when the first IBM Virtual Shared Disk is configured and the IBM Virtual Shared Disk device driver is loaded. Thereafter, sequence numbers are incremented as requests are sent to (outgoing) and received from (expected) other nodes, and reset via **ctlvsd −R** | **−r** commands.

Reloading the IBM Virtual Shared Disk device driver by **suspendvsd −a**, **stopvsd −a**, or **ucfgvsd −a** followed by **cfgvsd** also resets all sequence numbers to 0.

Initially, all nodes in the IBM Virtual Shared Disk network are cast in. The **ctlvsd −k** command casts a node out. The local node ignores requests from cast out nodes. The **ctlvsd −r** command casts nodes back in.

Files

/usr/lpp/csd/bin/ctlvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use the **–k** and **–K** options. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **statvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for information on tuning IBM Virtual Shared Disk performance and sequence numbers.

Examples

To display the current parameters, enter:

```
ctlvsd
```

The system displays a message similar to the following:

```
The current cache buffer count is 64.
The maximum cache buffer count is 256.
The request block count is 256.
The pbuf's count is 48.
The minimum buddy buffer size is 4096.
The maximum buddy buffer size is 65536.
The total buddy buffer size is 4 max buffers, 262144 bytes.
```

To display the mbuf headers and current routing table, enter:

```
ctlvsd -t
```

The system displays the following information:

```
Mbuf Cache Stats:
                Header
   Cached        1
      Hit     1023
     Miss        1

Route cache information:
 destination  interface  ref  status  direct/gateway   min managed mbuf
     1         css0        2    Up         Direct              256
```

defhsd

Purpose

defhsd – Defines a data striping device (HSD).

Syntax

defhsd **protect_LVCB** | **not_protect_LVCB** *hsd_name stripe_size vsd_name...*

Flags

None.

Operands

protect_lvcb | **not_protect_lvcb**
Protects the logical volume control block information that is stored at the first block of a logical volume. If **protect_lvcb** is specified, the data striping device (HSD) will skip the first stripe on each underlying IBM Virtual Shared Disk in an HSD. In this case, you should define each logical volume one stripe larger than necessary. If the IBM Virtual Shared Disk and Logical Volume Manager (LVM) disk mirroring are used, the logical volume control block information is critical.

hsd_name Specifies a unique name for the new HSD. This name must be unique across the system partition and should be unique across the SP to avoid any naming conflicts during future system partitioning operations. The length of the name must be less than or equal to 31 characters.

stripe_size Specifies the maximum size of data stored on an IBM Virtual Shared Disk at one time. The smallest stripe size is 4096 bytes. The stripe size must be a multiple of 4096 and less than or equal to 1GB.

vsd_name Specifies the IBM Virtual Shared Disks that compose the HSD. All underlying IBM Virtual Shared Disks in the HSD must be defined before using this command.

Description

The **defhsd** command is used to specify the *hsd_name*, *stripe size* and underlying IBM Virtual Shared Disks for the new data striping device (HSD).

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_data
```

and select the Define a Hashed Shared Disk option.

Files

/usr/lpp/csd/bin/defhsd Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **hsdatalst**, **undefhsd**

Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for information on tuning IBM Virtual Shared Disk performance and sequence numbers.

Examples

The following example adds SDR information indicating a stripe size of 32768, composed of *vsd.vsdn101*, *vsd.vsdn201*, and the name *hsd1* is defined.

```
defhsd hsd1 32768 vsd.vsdn101 vsd.vsdn201
```

defvsd

Purpose

defvsd – Defines an IBM Virtual Shared Disk.

Syntax

defvsd *logical_volume_name global_group_name vsd_name* [**nocache** | **cache**]

Flags

None.

Operands

logical_volume_name Is the name of the logical volume you want to specify as an IBM Virtual Shared Disk. This logical volume must reside on the global volume group indicated. The length of the name must be less than or equal to 15 characters.

global_group_name Is the name of the globally-accessible volume group previously defined by the **vsdvg** command where you want to specify an IBM Virtual Shared Disk. The length of the name must be less than or equal to 31 characters.

vsd_name Specifies a unique name for the new IBM Virtual Shared Disk. This name must be unique across the system partition and should be unique across the SP, to avoid any naming conflicts during future system partitioning operations. The suggested naming convention is **vsdnn***gvg_name*. The length of the name must be less than or equal to 31 characters.

Note: If you choose a *vsd_name* that is already the name of another device, the **cfgvsd** command will fail for that IBM Virtual Shared Disk. This failure ensures that the special device files created for the name do not overlay and destroy files of the same name representing some other device type (such as a logical volume).

nocache | **cache** Affects how requests are processed at the server node. **nocache** is the default. **cache** tells the IBM Virtual Shared Disk software on the server node to use the cache for all 4KB requests on 4KB boundaries. Otherwise, the cache is not used.

The **cache** option should only be used if the using application gains performance by avoiding a 4KB read immediately after a 4KB write. Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for additional information on IBM Virtual Shared Disk tuning. When using the **css0** adapter, there is no cache.

defvsd

Description

This command is run to specify logical volumes residing on globally accessible volume groups to be used as IBM Virtual Shared Disks.

You can use the System Management Interface Tool (SMIT) to run the **defvsd** command. To use SMIT, enter:

```
smit vsd_data
```

and select the Define a Virtual Shared Disk option.

Security

You must have root privilege to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdatalst**, **vsdvg**, **undefvsd**

Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for information regarding IBM Virtual Shared Disk performance enhancements.

Examples

1. The following example adds SDR information indicating that on globally accessible volume group **vg1n1**, the logical volume known as **lv1vg1n1** is used as a noncached IBM Virtual Shared Disk named **vsd1vg1n1**.

   ```
   defvsd lv1vg1n1 vg1n1 vsd1vg1n1
   ```

2. The following example defines cachable IBM Virtual Shared Disk **vsd1vg2n1** on the **lv2vg1n1** logical volume on the **vg1n1** globally accessible volume group

   ```
   defvsd lv2vg1n1 vg1n1 vsd1vg2n1 cache
   ```

delnimclient

Purpose

delnimclient – Deletes a Network Installation Management (NIM) client definition from a NIM master.

Syntax

delnimclient –**h** | –**l** *node_list* | –**s** *server_node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to be unconfigured as NIM clients of their boot/install servers. The *node_list* is a comma-separated list of node numbers.

–**s** *server_node_list*
Indicates by *server_node_list* the SP boot/install server nodes on which to delete all NIM clients that are no longer defined as boot/install clients in the System Data Repository (SDR). Server node 0 (zero) signifies the control workstation.

Operands

None.

Description

Use this command to undefine a node as a NIM client. This is accomplished by determining the node's boot/install server and unconfiguring that client node as a NIM client on that server. When complete, the entry for the specified client is deleted from the NIM configuration database on the server. This command does not change the boot/install attributes for the nodes in the System Data Repository.

Note: This command results in no processing on the client node.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/delnimclient

Related Information

Commands: **mknimclient**, **setup_server**

Examples

To delete the NIM client definition for nodes 1, 3, and 5 from the NIM database on their respective boot/install servers, enter:

```
delnimclient -l 1,3,5
```

delnimmast

Purpose

delnimmast – Unconfigures a node as a Network Installation Management (NIM) master.

Syntax

delnimmast –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to be unconfigured as NIM masters. The *node_list* is a comma-separated list of node numbers. Node number 0 (zero) signifies the control workstation.

Operands

None.

Description

Use this command to undefine a node as a NIM master. This command does not change the boot/install attributes for the nodes in the System Data Repository.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/delnimmast

Related Information

Commands: **mknimmast**, **setup_server**

Examples

To unconfigure nodes 1, 3, and 5 as NIM masters and delete the NIM file sets, enter:

```
delnimmast -l 1,3,5
```

dsh

Purpose

dsh – Issues commands to a group of hosts in parallel.

Syntax

dsh [–**q**]

dsh [–**h**]

dsh [–**i**] [**v**] [**c**] [**a**] [**G**] [–**l** *login_name*] [–**N** *node_group*,*node_group*, ...]
[–**w** {*host_names* | –}] [–**f** *fanout_value*] [*command*]

Flags

–**q** Displays a list of hosts in the current working collective file. The WCOLL environment variable is examined to find the name of the file containing the host names in a working collective, and host names from that file are displayed. In addition, the value of the FANOUT environment variable is displayed.

–**h** Displays usage information.

–**i** Contains information about the working collective and commands. If this flag is set, the working collective and the command is displayed as each command is issued.

–**v** Verifies hosts before adding to the working collective. If this flag is set, each host to be added to the working collective is checked before it is added to the collective. If a host is not responding, it is not included in the working collective.

–**c** Indicates that **dsh** continues to send commands to hosts for which previous **rsh**'s have returned a nonzero return code. If this flag is not set, the host is removed from the working collective for the duration of this **dsh** command.

–**a** Specifies that the System Data Repository **initial_hostname** field for all nodes in the current system partition be added to the working collective. If –**G** is specified, all nodes in the SP system are included.

–**G** Changes the scope of the –**a** and –**N** arguments from the current system partition to the SP system.

–**l** Specifies a remote user name under which to execute the commands. If **l** is not used, the remote user name is the same as your local user name. (This is lowercase **l**, as in **l**ist.)

–**w** Specifies a list of host names, separated by commas, to include in the working collective. Both this flag and the **a** flag can be included on the same command line. If "–" is specified, host names are read from standard input. If –**w** – is used, commands cannot be read from standard input. Duplicate host names are only included once in the working collective.

–**f** Specifies a fanout value. The default value is 64. It indicates the maximum number of concurrent **rsh**'s to execute. Sequential execution can be specified by indicating a fanout value of 1. The

fanout value is taken from the FANOUT environment variable if the **f** flag is not specified, otherwise the default is used.

–N Specifies a list of node groups. Each node group is resolved into nodes and these nodes are added to the working collective. If **–G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

If the **–a**, **–w**, or **–N** flags are not specified, the WCOLL environment variable contains the name of a file containing host names for the working collective.

Operands

command Specifies a command to execute on the working collective. It is passed to **rsh**. This command is specified in **rsh** syntax (see the SP **rsh** command).

Description

The **dsh** executes commands against all or any subset of the hosts in a network. It reads lines from the command line or standard input and executes each as a command on a set of network-connected hosts. These commands are in **rsh** syntax. Alternatively, a single command in **rsh** syntax can be specified on the **dsh** command line.

As each command is read, it is interpreted by passing it to each host in a group called the *working collective* via the SP **rsh** command.

The working collective is obtained from the first existence of one of the following:

1. A list of host names specified on the command line and the members of the cluster as listed in the System Data Repository.
2. The contents of a file named by the WCOLL environment variable.

If neither of these exist, an error has occurred and no commands are issued.

The working collective file should have one host name per line. Blank lines and comment lines beginning with **#** are ignored.

The path used when resolving the **dsh** command on the target nodes is the path set by the user with the DSHPATH environment variable. If DSHPATH is not set, the path used is the **rsh** default path, **/usr/ucb:/bin:/usr/bin:**. The DSHPATH environment variable only works when the user's remote login shell is the Bourne or Korn shell. An example would be to set DSHPATH to the path set on the source machine (for example, DSHPATH=$PATH).

The maximum number of concurrent **rsh**'s can be specified with the fanout (**f**) flag or via the FANOUT environment variable. If desired, sequential execution can be obtained by specifying a fanout value of 1. Results are displayed as remote commands complete. All **rsh**'s in a fanout must complete before the next set of **rsh**'s is started. If fanout is not specified via FANOUT or the **f** flag, **rsh**'s to 64 hosts are issued concurrently. Each **rsh** that **dsh** runs requires a reserved TCP/IP port and only 512 such ports are available per host. With large fanouts, it is possible to exhaust all the ports on a host, causing commands that use these ports, such as the SP **rlogin** and the SP **rsh** commands, to fail.

Exit values for the **rsh** commands are displayed in messages from **dsh** if nonzero. (A nonzero return code from **rsh** indicates that the **rsh** has failed; it has nothing to do with the exit code of the remotely executed command.) If an **rsh** fails, that host is removed from the current working collective (not the current working collective file), unless the **c** flag was set.

The **dsh** exit value is 0 if no errors occurred in the **dsh** command and all **rsh**'s finished with exit codes of 0. The **dsh** exit value is more than 0 if internal errors occur or the **rsh**'s fail. The exit value is increased by 1 for each **rsh** failure.

No particular error recovery for command failure on remote hosts is provided. The application or user can examine the command results in **dsh**'s standard error and standard output, and take appropriate action.

The **dsh** command waits until results are in for each command for all hosts and displays those results before reading more input commands.

The **dsh** command does not work with interactive commands, including those that read from standard input.

The **dsh** command output consists of the output (standard error and standard output) of the remotely executed commands. The **dsh** standard output is the standard output of the remote command. The **dsh** standard error is the standard error of the remote command. Each line is prefixed with the host name of the host from which that output came. The host name is followed by ":" and a line of the command output.

For example, let's say that a command was issued to a working collective of host1, host2, and host3. When the command was issued on each of the hosts, the following lines were written by the remote commands:

```
For host1 stdout:
h1out1
h1out2

For host2 stdout:
h2out1
h2out2

For host3 stdout:
h3out1

For host3 stderr:
h3err1
h3err2

dsh stdout will be
host1: h1out1
host1: h1out2
host2: h2out1
host2: h2out2
host3: h3out1

dsh stderr will be
host3: h3err1
host3: h3err2
```

A filter to display the output lines by the host is provided separately. See the **dshbak** command.

If a host is detected as down (for example, an **rsh** returns a nonzero return code), subsequent commands are not sent to it on this invocation of **dsh**, unless the **c** (continue) option is specified on the command line.

An exclamation point at the beginning of a command line causes the command to be passed directly to the local host in the current environment. The command is not sent to the working collective.

Signals 2 (INT), 3 (QUIT), and 15 (TERM) are propagated to the remote commands.

Signals 19 (CONT), 17 (STOP), and 18 (TSTP) are defaulted. This means that the **dsh** command responds normally to these signals, but they do not have an effect on the remotely running commands. Other signals are caught by **dsh** and have their default effects on the **dsh** command. In the case of these other signals, all current children, and via propagation their remotely running commands, are terminated (SIGTERM).

Security considerations are the same as for the SP **rsh** command.

Files

/usr/sbin/dsh The **dsh** command.

/usr/sbin/dshbak
The supplied backend formatting filter.

working collective file
A file containing host names, one per line, that defines a working collective.

Related Information

Command: **dshbak**

SP Commands: **rsh**, **sysctl**

Examples

1. To issue the **ps** command on each host listed in the **wchosts** file, enter:

   ```
   WCOLL=./wchosts dsh ps
   ```

2. To list the current working collective file as specified by the WCOLL environment variable, enter:

   ```
   dsh -q
   ```

3. To set the working collective to three hosts and start reading commands from standard input, enter:

   ```
   dsh -w otherhost1,otherhost2,otherhost3
   ```

4. To set the current working collective to three hosts, plus the members of the cluster, and issue a command on those hosts formatting the output, enter:

   ```
   dsh -w host1,host2,host3 -a cat /etc/passwd | dshbak
   ```

5. To append the file remotefile on otherhost to otherremotefile, which is on otherhost, enter:

   ```
   dsh -w otherhost cat remotefile ">>" otherremotefile
   ```

6. To run a file of commands sequentially on all the members of the current system partition and save the results in a file, including the collective and the working collective for each command, enter:

   ```
   dsh -if 1 -a < commands_file > results 2>&1
   ```

7. To run the **ps** command on the working collective and filter results locally, enter:

   ```
   dsh ps -ef | grep root
   ```

8. To run the **ps** command and filter results on the working collective hosts (this can improve performance considerably), enter:

   ```
   dsh 'ps -ef | grep root'
   ```

   or

   ```
   dsh ps -ef "|" grep root
   ```

9. To cat a file from host1 to the local system stripping off the preceding host name to preserve the file, enter:

   ```
   dsh -w host1 cat /etc/passwd | cut -d: -f2- | cut -c2- > myetcpasswd
   ```

10. To run the **ps** command on each node in the node group my_nodes, enter:

    ```
    dsh -N my_nodes ps
    ```

dshbak

Purpose

dshbak – Presents formatted output from the **dsh** and **sysctl** commands.

Syntax

dshbak [–**c**]

Flags

–**c** Collapses identical output from more than one host so that it is displayed only once.

Operands

None.

Description

The **dshbak** command takes lines in the following format:

```
host_name: line of output from remote command
```

The **dshbak** command formats them as follows and writes them to standard output. Assume that the output from host_name3 and host_name4 is identical and the **c** option was specified:

```
HOSTS -------------------------------------------------------------
host_name1
-------------------------------------------------------------------
.
.
lines from dsh or sysctl with host_names stripped off
.
.

HOSTS -------------------------------------------------------------
host_name2
-------------------------------------------------------------------
.
.
lines from dsh or sysctl with host_names stripped off
.
.

HOSTS -------------------------------------------------------------
host_name3             host_name4
-------------------------------------------------------------------
.
.
lines from dsh or sysctl with host_names stripped off
.
.
```

When output is displayed from more than one host in collapsed form, the host names are displayed alphabetically.

When output is not collapsed, output is displayed sorted alphabetically by host name.

The **dshbak** command writes "." for each 1000 lines of output filtered.

Files

/usr/sbin/dshbak
The **dshbak** command.

Related Information

Commands: **dsh**, **sysctl**

Examples

1. To display the results of a command executed on several hosts in the format described previously, enter:

   ```
   dsh -w host1,host2,host3 cat /etc/passwd | dshbak
   ```

2. To display the results of a command executed on several hosts with identical output displayed only once, enter:

   ```
   dsh -w host1,host2,host3 pwd | dshbak -c
   ```

Eannotator

Purpose

Eannotator – Annotates the connection labels in the topology file.

Syntax

Eannotator –**F** *input_file* –**f** *output_file* –**O** [**yes** | **no**]

Flags

–**F** Specifies the topology input file.

–**f** Specifies the topology output file.

–**O** Specifies whether to save the output file to the System Data Repository (SDR) or to the current directory. **yes** saves the output file to the SDR via the **Etopology** command. **no** saves the output file to the current directory.

Operands

None.

Description

This command supports all of the following:

- The High Performance Switch
- The High Performance Switch-LC8
- The SP Switch
- The SP Switch-8

This command must be executed whenever a new topology file is selected.

The topology file, which describes the wiring configuration for the High Performance Switch, contains node-to-switch or switch-to-switch cable information. A node-to-switch connection looks like following:

```
s 25 tb0 17 0     E2-S00-BH-J16 to E2-N2
```

The connection label “E2-S00-BH-J16 to E2-N2” provides physical connection information for a customer’s use to identify the bad connection.

Depending on the type of switch installed (High Performance Switch or SP Switch), together with the customer’s physical switch frame configuration defined in the SDR, the **Eannotator** command retrieves Switch and Node objects from the SDR and applies proper connection information to the topology file.

If the input topology file contains existing connection information, the **Eannotator** command replaces the existing connection label with the new connection labels. If the input topology file does not contain connection labels, the **Eannotator** command appends the proper connection label to each line on the topology file.

Files

/etc/SP/expected.top.1nsb_8.0isb.0
The standard topology file for systems with the 8-port High Performance Switch or a maximum of eight nodes.

/etc/SP/expected.top.1nsb.0isb.0
The standard topology file for one NSB system or a maximum of 16 nodes.

/etc/SP/expected.top.2nsb.0isb.0
The standard topology file for two NSB systems or a maximum of 32 nodes.

/etc/SP/expected.top.3nsb.0isb.0
The standard topology file for three NSB systems or a maximum of 48 nodes.

/etc/SP/expected.top.4nsb.0isb.0
The standard topology file for four NSB systems or a maximum of 64 nodes.

/etc/SP/expected.top.5nsb.0isb.0
The standard topology file for five NSB systems or a maximum of 80 nodes.

/etc/SP/expected.top.5nsb.4isb.0
The standard topology file for five NSB and four ISB systems or a maximum of 80 nodes. This is an advantage-type network with a higher bisectional bandwidth.

/etc/SP/expected.top.6nsb.4isb.0
The standard topology file for six NSB and four ISB systems or a maximum of 96 nodes.

/etc/SP/expected.top.7nsb.4isb.0
The standard topology file for seven NSB and four ISB systems or a maximum of 112 nodes.

/etc/SP/expected.top.8nsb.4isb.0
The standard topology file for eight NSB and four ISB systems or a maximum of 128 nodes.

/etc/SP/expected.top.1nsb_8.0isb.1
The standard topology file for systems with an SP Switch-8 and a maximum of eight nodes.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eclock**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eupartition**

Refer to *IBM Parallel System Support Programs for AIX: System Planning* for details about system partition topology files.

Examples

1. The following are examples of topology file entries.

   The precoded connection labels on the topology file start with an L which indicates logical frames. The **Eannotator** command replaces the character L with an E which indicates physical frames. The S character indicates which slot the switch occupies in the frame, the BH characters indicate a Bulk Head connection, the J character indicates which jack provides the connection from the switch board, and the N character indicates the node being connected to the switch.

   The following are the topology file entries before and after the **Eannotator** command executes:

   ```
   Before:
   s 15 3  tb0 0 0     L01-S00-BH-J18 to L01-N1

   After:
   s 15 3  tb0 0 0     E01-S17-BH-J18 to E01-N1
   ```

   Note: Logical frame L01 is defined as physical frame 1 in the SDR Switch object.

   ```
   Before:
   s 10016 0 s 51 3     L09-S1-BH-J20 to L05-S00-BH-J19

   After:
   s 10016 0 s 51 3     E10-S1-BH-J20 to E05-S17-BH-J19
   ```

   Note: Logical frame L09 is defined as physical frame 10 in the SDR Switch object.

2. To annotate a topology file for a 128-way SP system with eight Node Switch Boards (NSBs) and four Intermediate Switch Boards (ISBs) and to save the output file in the current directory, enter:

   ```
   Eannotator -F expected.top.8nsb.4isb.0 -f expected.top -O no
   ```

3. To annotate a topology file for a 16-way SP system with one NSB and no ISBs and to save the output file in the SDR via the **Etopology** command, enter:

   ```
   Eannotator -F expected.top.1nsb.0isb.0 -f expected.top -O yes
   ```

Eclock

Purpose

Eclock – Controls the clock source for each switch board within an SP cluster.

Syntax

Eclock [–**f** *Eclock_topology_file*] | [–**a** *Eclock_topology_file*] | [–**r**] | [–**d**] | [–**s** *switch_number* –**m** *mux_value*] | [–**c** *Eclock_topology_file*]

Flags

–**f** *Eclock_topology_file*
: Specifies the file name of the clock topology file containing the initial switch clock input values for all switches in the system.

–**a** *Eclock_topology_file*
: Uses the alternate **Eclock** topology specified in the given clock topology file.

–**r**
: Extracts the clock topology file information from the System Data Repository (SDR) and initializes the switch clock inputs for all switches in the system.

–**d**
: Detects the switch configuration, automatically selects the clock topology file, and initializes the switch clock inputs for all switches in the system.

–**s** *switch_number* –**m** *mux_value*
: Sets an individual switch (*switch_number*) clock multiplexor (mux) value (*mux_value*)

  where:

  switch_number Specifies the switch number.

  mux_value Specifies a flag with one of the following values:

  High Performance Switch

  0 Use the internal oscillator (make this frame the master frame).
  1 Use input 1 (clock input from jack 3).
  2 Use input 2 (clock input from jack 5).
  3 Use input 3 (clock input from jack 7).

  SP Switch

  0 Use the internal oscillator (make this frame the master frame).
  1 Use input 1 (clock input from jack 3).
  2 Use input 2 (clock input from jack 4).
  3 Use clock input from jack 5 (NSBs or ISBs).
  4 Use clock input from jack 4 (NSBs or ISBs).
  5 Use clock input from jack 5 (NSBs or ISBs).
  6 Use clock input from jack 6 (NSBs or ISBs).
  7 Use clock input from jack 7 (ISBs only).
  8 Use clock input from jack 8 (ISBs only).
  9 Use clock input from jack 9 (ISBs only).

10 Use clock input from jack 10 (ISBs only).
27 Use clock input from jack 27 (NSBs or ISBs).
28 Use clock input from jack 28 (NSBs or ISBs).
29 Use clock input from jack 29 (NSBs or ISBs).
30 Use clock input from jack 30 (NSBs or ISBs).
31 Use clock input from jack 31 (ISBs only).
32 Use clock input from jack 32 (ISBs only).
33 Use clock input from jack 33 (ISBs only).
34 Use clock input from jack 34 (ISBs only).

–**c** *Eclock_topology_file*
Creates a new clock topology file from the data in the SDR.

If a flag is not specified, the clock input values stored in the SDR are displayed.

Operands

None.

Description

Use this command to set the multiplexors that control the clocking at each switch board within the configuration. One switch board within the configuration is designated as the “Master” switch that provides the clocking signal for all other switch boards within the configuration. The **Eclock** command reads clock topology information from either the file specified on the command line or the clock topology data within the SDR. If a clock topology file was specified, the **Eclock** command places the clock topology information into the SDR, so that it can be accessed again during a subsequent **Eclock** invocation. After processing the clock topology file, **Eclock** causes the new clock topology to take effect for the switches specified. A clock topology file contains the following information for each switch board within the cluster:

- A switch board number.
- A flag indicating the clock source for the switch within the frame. The flag can have any of the following settings:

  High Performance Switch

  0 Use the internal oscillator (make this frame the master frame).
  1 Use input 1 (clock input from jack 3).
  2 Use input 2 (clock input from jack 5).
  3 Use input 3 (clock input from jack 7).

  SP Switch

  0 Use the internal oscillator (make this frame the master frame).
  1 Use input 1 (clock input from jack 3).
  2 Use input 2 (clock input from jack 5).
  3 Use clock input from jack 3 (NSBs or ISBs).
  4 Use clock input from jack 4 (NSBs or ISBs).
  5 Use clock input from jack 5 (NSBs or ISBs).
  6 Use clock input from jack 6 (NSBs or ISBs).
  7 Use clock input from jack 7 (ISBs only).
  8 Use clock input from jack 8 (ISBs only).
  9 Use clock input from jack 9 (ISBs only).
  10 Use clock input from jack 10 (ISBs only).
  27 Use clock input from jack 27 (NSBs or ISBs).

28 Use clock input from jack 28 (NSBs or ISBs).
29 Use clock input from jack 29 (NSBs or ISBs).
30 Use clock input from jack 30 (NSBs or ISBs).
31 Use clock input from jack 31 (ISBs only).
32 Use clock input from jack 32 (ISBs only).
33 Use clock input from jack 33 (ISBs only).
34 Use clock input from jack 34 (ISBs only).

- The receiving jack number (*xx* if this switch board is the master).
- The switch board number providing the clocking signal (zero if this switch board is the master).
- The clock source jack number.
- An alternate clock source (mux) value.
- The alternate clock receiver jack number.
- The switch board number providing the alternate clock source (mux) value.
- The alternate sending jack number (*xx* if this switch board is the master).

High Performance Switch Warning

Warning: do not change the switch clock multiplexor settings (with **Eclock**, **spmon** command/GUI, hmcmds) while the nodes are powered on. Otherwise, AIX must be rebooted and **Estart** must be run following the clock adjustment.

SP Switch Warning

Warning: do not change the switch clock multiplexor settings (with **Eclock**, **spmon** command/GUI, hmcmds) while the nodes are powered on. Otherwise, **Estart** must be run following the clock adjustment.

To execute the **Eclock** command, the user must be authorized to access the Hardware Monitor subsystem and, for those frames specified to the command, the user must be granted **VFOP** (Virtual Front Operator Panel) permission. Commands sent to frames for which the user does not have **VFOP** permission are ignored. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **kinit** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

Files

/etc/SP/Eclock.top.1nsb.0isb.0
The standard clock topology file for systems with one NSB or a maximum of 16 nodes.

/etc/SP/Eclock.top.1nsb_8.0isb.0
The standard clock topology file for systems with the 8-port High Performance Switch or an SP Switch-8 or a maximum of eight nodes.

/etc/SP/Eclock.top.2nsb.0isb.0
The standard clock topology file for systems with two NSBs or a maximum of 32 nodes.

/etc/SP/Eclock.top.3nsb.0isb.0
The standard clock topology file for systems with three NSBs or a maximum of 48 nodes.

/etc/SP/Eclock.top.4nsb.0isb.0
The standard clock topology file for systems with four NSBs or a maximum of 64 nodes.

/etc/SP/Eclock.top.5nsb.0isb.0
The standard clock topology file for systems with five NSBs or a maximum of 80 nodes.

/etc/SP/Eclock.top.5nsb.4isb.0
The standard clock topology file for systems with five NSBs and four ISBs or a maximum of 80 nodes. This is an advantage-type network with a higher bisectional bandwidth.

/etc/SP/Eclock.top.6nsb.4isb.0
The standard clock topology file for systems with six NSBs and four ISBs or a maximum of 96 nodes.

/etc/SP/Eclock.top.7nsb.4isb.0
The standard clock topology file for systems with seven NSBs and four ISBs or a maximum of 112 nodes.

/etc/SP/Eclock.top.8nsb.4isb.0
The standard clock topology file for systems with eight NSBs and four ISBs or a maximum of 128 nodes.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eunpartition**

Examples

1. To set the clock multiplexors for a 128-way SP system with eight Node Switch Boards (NSBs) and four Intermediate Switch Boards (ISBs), enter:

   ```
   Eclock -f /etc/SP/Eclock.top.8nsb.4isb.0
   ```

2. To display the clock multiplexor settings for all switches within the SP system, enter:

   ```
   Eclock
   ```

3. To set the switch on frame 1 (switch 1) to be the master switch (use internal oscillator), enter:

   ```
   Eclock -s 1 -m 0
   ```

4. To create an **Eclock** topology file from the current data in the SDR, enter:

   ```
   Eclock -c /tmp/Eclock.top
   ```

5. To use an alternate clock topology (with a new switch clock source) for a 64-way SP system with two ISBs, enter:

   ```
   Eclock -a /etc/SP/Eclock.top.4nsb.2isb.0
   ```

6. To have **Eclock** automatically select a topology file for you based on data in the SDR, enter:

   ```
   Eclock -d
   ```

Eduration

Purpose

Usage Note

Do not use this command if you have the SP Switch installed on your system.

Eduration – Sets the interval that nodes can be added or removed from the High Performance Switch. This interval is called the Run Phase Duration.

Syntax

Eduration [[*days* **day**[**s**]] [*hours* **hour**[**s**]] [*minutes* **minute**[**s**]]] | [–**h**]

Flags

days **day[s]** Specifies the number of days that the switch will stay in the Run Phase. Valid values are 1—40.

hours **hour[s]** Specifies the number of hours that the switch will stay in the Run Phase. Valid values are 1—23.

minutes **minute[s]**
Specifies the number of minutes that the switch will stay in the Run Phase. Valid values are 1—59.

–**h** Displays usage information.

Any combination of the three preceding time designations can be used to specify the new Run Phase Duration. Since the duration determines how quickly the system can respond to **Efence** and **Eunfence** requests, it should be set to provide the desired response. If none of the time specifiers are present, **Eduration** will display the current value of the Run Phase Duration.

Operands

None.

Description

The Run Phase Duration controls how frequently **Efence** and **Eunfence** requests are handled. This command provides an interface to set that value.

Note: The Run Phase Duration changes will not take effect until the end of the current Run Phase. If you are changing the Run Phase Duration from a large value to something that is significantly smaller and you do not want to wait for the current the Run Phase to complete, you will have to **Estart** the switch.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eunpartition**

Examples

1. To set the Run Phase Duration to 1 minute, enter:

   ```
   Eduration 1 minute
   ```

2. To set the Run Phase Duration to an hour and 30 minutes, enter:

   ```
   Eduration 1 hour 30 minutes
   ```

3. To query the Run Phase Duration, enter:

   ```
   Eduration
   ```

Efence

Purpose

Efence – Removes an SP node from the current *active* switch network.

Syntax

Efence [–**h**] | [–**G**] [–**autojoin**] [*node_specifier*] ...

Flags

–**h** Displays usage information.

–**G** Fences all valid nodes in the list of nodes regardless of system partition boundaries. If the –**G** flag is not used, the **Efence** command will only fence the nodes in the current system partition. All other specified nodes will not be fenced and a nonzero return code is returned.

–**autojoin** Enables the nodes in the argument list to be fenced and to automatically rejoin the current switch network if the node is rebooted or the Fault Service daemon is restarted.

If you have an SP Switch installed on your system, such nodes are also rejoined when an **Estart** command is issued.

Operands

node_specifier

Specifies a node or a list of nodes that are to be taken out of the current switch network. It can be a list of host names, IP addresses, node numbers, frame,slot pairs, or a node group.

If nodes are not specified, a list of nodes that were fenced from the current switch network will be displayed. If –**autojoin** is used without a node list, only the nodes that were fenced with **autojoin** enabled will be displayed.

Note: You cannot fence either the primary node on the High Performance Switch or the primary backup node on the SP Switch.

Description

Use this command to fence a node from the current switch network.

If you have an SP Switch installed on your system, you must do either of the following to bring the node back up onto the switch network:

- Specify the –**autojoin** option so that the node will be rejoined on a reboot, a restart of the Fault Service Daemon, or a switch network **Estart**.
- Specify the **Eunfence** command.

If you have a High Performance Switch installed on your system, you can issue the **Estart** command to rejoin all nodes on the switch network.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eunpartition**

Examples

1. To display all the nodes that were fenced from the switch network in the current system partition, enter:

   ```
   Efence
   ```

2. To display only the nodes that were fenced from the switch network with the automatic join option enabled, enter:

   ```
   Efence -autojoin
   ```

3. To display all the nodes that were fenced from the switch network in all system partitions, enter:

   ```
   Efence -G
   ```

4. To fence two nodes by IP address, enter:

   ```
   Efence 129.33.34.1 129.33.34.6
   ```

5. To fence a node by host name, enter:

   ```
   Efence r11n01
   ```

6. To fence a list of nodes by node number and enable –**autojoin**, enter:

   ```
   Efence -autojoin 54 65 32 78
   ```

7. To fence node 14 of frame 2 by frame,slot pair, enter:

   ```
   Efence 2,14
   ```

8. If the current partition has nodes with node numbers 1, 2, 5, and 6 and another partition has nodes with node numbers 3, 4, 7, and 8, issuing the command:

   ```
   Efence 5 6 7 8
   ```

   fences nodes 5 and 6, but not nodes 7 and 8. As a result, the command returns a nonzero return code.

9. To successfully fence the nodes in example 8 with the same partitions, use the –**G** flag as follows:

   ```
   Efence -G 5 6 7 8
   ```

emconditionctrl script

Purpose

emconditionctrl – Loads the System Data Repository (SDR) with predefined Event Management conditions.

Syntax

emconditionctrl [–**a**] [–**s**] [–**k**] [–**d**] [–**c**] [–**t**] [–**o**] [–**r**] [–**h**]

Flags

–**a**	Loads the SDR with predefined Event Management conditions for the current system partition.
–**s**	Starts the subsystem. (Currently has no effect.)
–**k**	Stops the subsystem. (Currently has no effect.)
–**d**	Deletes the subsystem. (Currently has no effect.)
–**c**	Cleans the subsystem. (Currently has no effect.)
–**t**	Turns tracing on. (Currently has no effect.)
–**o**	Turns tracing off. (Currently has no effect.)
–**r**	Refreshes the subsystem. (Currently has no effect.)
–**h**	Displays usage information.

Operands

None.

Description

The **emconditionctrl** script loads the SDR with some useful conditions that can be used for registering for Event Management events. Currently the SP Perspectives application can make use of conditions.

The **emconditionctrl** script is not normally executed on the command line. It is normally called by the **syspar_ctrl** command after the control workstation has been installed or when the system is partitioned. It implements all of the flags that **syspar_ctrl** can pass to its subsystems, although only the –**a** flag causes any change to the system. The –**a** flag causes predefined conditions to be loaded only if run on the control workstation. It has no effect if run elsewhere.

Exit Values

0	Indicates the successful completion of the command.
nonzero	Indicates an exit code from the **SDRCreateObjects** command.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/emconditionctrl

Related Information

Commands: **syspar_ctrl**

emonctrl script

Purpose

emonctrl – A control script that manages the Emonitor subsystem.

Syntax

emonctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem. Not implemented. The subsystem should be started using **Estart -m**
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystem. Not used.
–**o**	Turns tracing off for the subsystem. Not used.
–**r**	Refreshes the subsystem. Not implemented.
–**h**	Displays usage information.

Operands

None.

Description

The Emonitor subsystem monitors designated nodes in an attempt to maximize their availability on the switch network.

The **emonctrl** control script controls the operation of the Emonitor subsystem. The subsystem is under the control of the System Resource Controller (SRC) and belongs to a subsystem group called **emon**.

An instance of the Emonitor subsystem can execute on the control workstation for each system partition. Because **Emonitor** provides its services within the scope of a system partition, it is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It should be issued from the control workstation and is not functional on the nodes.

From an operational point of view, the Emonitor subsystem group is organized as follows:

Subsystem	Emonitor
Subsystem Group	**emon**

SRC Group	**emon** The **emon** group is associated with the **Emonitor** daemon. On the control workstation, there are multiple instances of **Emonitor**, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **Emonitor.sp_prod** and **Emonitor.sp_test**.
Daemons	**Emonitor** The **Emonitor** daemon provides switch node monitoring.

The **emonctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **emonctrl** script provides a variety of controls for operating the **Emonitor** daemon:

- Adding, stopping, and deleting the subsystem
- Cleaning up the subsystems, that is, deleting them from all system partitions

Before performing any of these functions, the script obtains the current system partition name and IP address (using the **spget_syspar** command) and the node number (using the **node_number**) command. If the node number is zero, the control script is running on the control workstation. Since the **Emonitor** daemon runs only on the control workstation, the script performs no function when run on a node.

Except for the clean function, all functions are performed within the scope of the current system partition.

Adding the Subsystem

When the –**a** flag is specified, the control script uses the **mkssys** command to add the **Emonitor** daemon to the SRC. The control script operates as follows:

1. It checks whether the Emonitor subsystem already exists in this system partition. If the Emonitor subsystem does exist, it exits.
2. It adds the Emonitor subsystem to the SRC with the system partition name appended.

Starting the Subsystem

This option is unused since the **Emonitor** daemon must be started via **Estart -m**.

Stopping the Subsystem

When the –**k** flag is specified, the control script uses the **stopsrc** command to stop the **Emonitor** daemon in the current system partition.

Deleting the Subsystem

When the –**d** flag is specified, the control script uses the **rmssys** command to remove the Emonitor subsystem from the SRC. The control script operates as follows:

1. It makes sure that the **Emonitor** subsystem is stopped.
2. It removes the **Emonitor** subsystem from the SRC using the **rmssys** command.

Cleaning Up the Subsystems

When the –**c** flag is specified, the control script stops and removes the Emonitor subsystems for all system partitions from the SRC. The control script operates as follows:

1. It stops all instances of subsystems in the subsystem group in all system partitions, using the **stopsrc -g emon** command.
2. It removes all instances of subsystems in the subsystem group in all system partitions from the SRC using the **rmssys** command.

Turning Tracing On

Not currently used.

Turning Tracing Off

Not currently used.

Refreshing the Subsystem

Not currently used.

Logging

While it is running, the **Emonitor** daemon provides information about its operation and errors by writing entries in a log file. The **Emonitor** daemon uses log files called **/var/adm/SPlogs/css/Emonitor.log** and **/var/adm/SPlogs/css/Emonitor.Estart.log**.

Files

/var/adm/SPlogs/css/Emonitor.log
Contains the log of all **Emonitor** daemons on the system.

/var/adm/SPlogs/css/Emonitor.Estart.log
Contains the log of all **Estart** and **Eunfence** commands issued by all **Emonitor** daemons.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/emonctrl

Related Information

Commands: **Emonitor**, **Estart**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Emonitor subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   emonctrl -a
   ```

2. To stop the Emonitor subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   emonctrl -k
   ```

3. To delete the Emonitor subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   emonctrl -d
   ```

4. To clean up the Emonitor subsystem on all system partitions, enter:

   ```
   emonctrl -c
   ```

5. To display the status of all of the subsystems in the Emonitor SRC group, enter:

   ```
   lssrc -g emon
   ```

6. To display the status of an individual Emonitor subsystem, enter:

   ```
   lssrc -s subsystem_name
   ```

7. To display the status of all of the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

Emonitor daemon

Purpose

Emonitor – Monitors nodes listed in the /**etc**/**SP**/**Emonitor.cfg** file in an to attempt to maximize this availability on the switch.

Syntax

Emonitor

Flags

None.

Operands

None.

Description

Emonitor is a daemon controlled by the System Resource Controller (SRC). It can be used to monitor nodes in a system partition in regard to the their status on the switch. A system-wide configuration file (/**etc**/**SP**/**Emonitor.cfg**) lists all nodes on the system to be monitored. The objective is to bring these nodes back up on the switch network when necessary.

Emonitor is invoked with **Estart -m**. Once invoked, it is controlled by SRC so it will restart if it is halted abnormally. If the you decide to end monitoring, you must run /**usr**/**lpp**/**ssp**/**bin**/**emonctrl -s** to stop the daemon in your system partition.

There is an **Emonitor** daemon for each system partition. The daemon watches for any node coming up (for example, host_responds goes from 0 to 1). When the daemon detects a node coming up, it performs a review of the nodes in the configuration file to check if any node is off the switch network. If any nodes in the specified system partition are off the switch network, it determines a way to bring them back onto the the switch (for example, via **Eunfence** or **Estart**), and takes the appropriate action. In order to avoid the **Estart** command from being run several times (which can occur if multiple nodes are coming up in sequence), **Emonitor** waits 3 minutes after a node comes up to be sure no other nodes are in the process of coming up. Each time a new node comes up prior to the 3 minute timeout, **Emonitor** resets the timer to a maximum wait of 12 minutes.

Emonitor cannot always bring nodes back on the switch. For example, if any of the following occur:

- The fault service daemon is not operational on the node
- The primary node is down
- There is a hardware error

On a High Performance Switch, if a node is faulted off the switch and you are forced to do an **Estart**, you will lose history of any nodes that you had isolated off the switch. All nodes on a High Performance Switch come back on the switch on an **Estart**.

Problems can occur if the node that is faulted off the switch is experiencing a recurring error that causes it to come up and then fail repeatedly. The monitor continually attempts to bring this node into the switch network and could jeopardize the stability of the remaining switch network.

Files

/etc/SP/Emonitor.cfg
Specifies a list of node numbers, one per line, that the user wants monitored by **Emonitor**. This list is system-wide.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **emonctrl**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eupartition**

Eprimary

Purpose

Eprimary – Assigns or queries the switch primary node and switch primary backup node for a system partition.

High Performance Switch

Syntax

Eprimary [–**h**] [–**init**] [*node_identifier*]

Flags

–**h** Displays usage information.

–**init** Initializes or reinitializes the current system partition object. If –**init** is specified without a node identifier, the lowest numbered node in the system partition is used by default.

Operands

node_identifier
Specifies the node designated as the switch primary node. It can be a host name, an IP address, a frame,slot pair, or a node number.

Note: If no flags or operands are specified, the current switch primary node is displayed.

Description

Use this command to assign, change, or query the switch primary node. When the –**init** option is specified, it can be used to create a switch partition object for a system partition. The primary node should not be changed unless the current primary node is becoming unavailable (for example, if the current primary node is to be serviced). The **Estart** command must be issued before a change of the primary node (using **Eprimary**) takes effect. The old primary node must be rebooted or powered off before issuing **Estart** to remove its inclination to behave as the primary node.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**

Examples

1. To query the switch primary node, enter:

   ```
   Eprimary
   ```

2. To designate a switch primary node by IP address, enter:

   ```
   Eprimary 129.33.34.1
   ```

3. To designate a switch primary node by node number, enter:

   ```
   Eprimary 1
   ```

4. To designate the switch primary node by host name, enter:

   ```
   Eprimary r11n01
   ```

5. To create a system partition object and assign a switch primary node by a frame,slot, enter:

   ```
   Eprimary -init 1,2
   ```

SP Switch

Syntax

Eprimary [–**h**] [–**init**] [*node_identifier*] [–**backup** *bnode_identifier*]

Flags

–**h** Displays usage information.

–**init** Initializes or reinitializes the current system partition object. If –**init** is specified without a *node_identifier* or without a *bnode_identifier*, the respective default is used for the primary and primary backup nodes. The lowest numbered node in the system partition is the default primary node, and the furthest node from the primary is the default primary backup node.

–**backup** *bnode_identifier*
Specifies the node designated as the oncoming switch primary backup node. It can be a host name, an IP address, a frame,slot pair, or a node number. If *node_identifier* is not specified, a primary backup node is automatically selected.

Operands

node_identifier
Specifies the node designated as the oncoming switch primary node. It can be a host name, an IP address, a frame,slot pair, or a node number. If *node_identifier* is not specified, an oncoming primary node is automatically selected.

Note: If no flags or operands are specified, each of the following is displayed:

- Current switch primary node
- Current switch primary backup node
- Oncoming switch primary node
- Oncoming switch primary backup node

Description

Use this command to assign, change, or query the switch primary node or the switch primary backup node. The primary node should not be changed unless the current primary node is becoming unavailable (for example, if the current primary node is to be serviced). The **Estart** command must be issued before a change of the primary node or the primary backup node (using **Eprimary**) takes effect.

In an SP Switch network, the primary node takeover facility automatically handles situations (such as a node loss) for each of the primary and primary backup nodes. The primary node replaces a failing primary backup node and the primary backup node automatically takes over for the primary node if the primary node becomes unavailable. The primary backup node should be selected using the following guidelines:

- Specify a node from a different frame other than the primary node, if possible.
- If this node is not selected from a frame that is different from the primary node, it should be a node connected to a switch chip that is different from the primary node's. For example:
  - Node slots 1, 2, 5, and 6 are connected to the same switch chip.
  - Node slots 3, 4, 7, and 8 are connected to the same switch chip.
  - Node slots 9, 10, 13, and 14 are connected to the same switch chip.
  - Node slots 11, 12, 15, and 16 are connected to the same switch chip.

The **Eprimary** command selects a default oncoming primary or oncoming backup primary node if one is not specified. Users receive a warning in the following situations on the oncoming primary or oncoming backup primary nodes:

- If they cannot ping
- If the switch daemon is not up
- If the node is fenced

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**, **Eunpartition**

Examples

1. To query the switch primary and primary backup nodes, enter:

   ```
   Eprimary
   ```

2. To designate an oncoming switch primary node by IP address and let **Eprimary** select an oncoming switch primary backup node, enter:

   ```
   Eprimary 129.33.34.1
   ```

3. To designate an oncoming switch primary node and an oncoming switch primary backup node by IP address, enter:

   ```
   Eprimary 129.33.34.1 -backup 129.33.34.56
   ```

4. To designate an oncoming switch primary node and an oncoming switch primary backup node by host name, enter:

   ```
   Eprimary r11n01 -backup r17n02
   ```

5. To create a system partition object and assign a switch primary backup node by a frame,slot, enter:

   ```
   Eprimary -init 1,2 -backup 1,6
   ```

Equiesce

Purpose

Usage Note

Use this command only if you have an SP Switch installed on your system.

Equiesce – Quiesces the switch by causing the primary and primary backup nodes to shut down switch recovery and primary node takeover.

Syntax

Equiesce [–**h**]

Flags

–**h** Displays usage information.

Operands

None.

Description

Use this command to disable switch error recovery and primary node takeover. It is used to shut down normal switch error actions when global activities affecting nodes are performed. For example, when all nodes are shutdown or rebooted, they are fenced from the switch by the primary node.

If the primary node is not the first node to shut down during a global shutdown or reboot of the entire system, it may fence all the other nodes including the primary backup node. Primary node takeover can also occur if the primary node is shut down and the backup node remains up. Issuing the **Equiesce** command before the shutdown prevents these situations from occurring.

The **Equiesce** command causes the primary and primary backup nodes to shut down their recovery actions. Data still flows over the switch, but no faults are serviced and primary node takeover is disabled. Only the **Eannotator**, **Eclock**, **Eprimary**, **Estart**, and **Etopology** commands are functional after the **Equiesce** command is issued.

Estart must be issued when the global activity is complete to reestablish switch recovery and primary node takeover.

Security

You must have root privilege to run this command.

Location

/usr/lpp/ssp/bin/Equiesce

Related Information

Commands: **Eannotator**, **Eclock**, **Efence**, **Eprimary**, **Estart**, **Etopology**, **Eunfence**, **Eunpartition**

Examples

To quiesce the switch before shutting down the system, enter:

```
Equiesce
```

Estart

Purpose

Estart – Starts the High Performance Switch.

Syntax

Estart [–**h**] [–**m**]

Flags

–**h** Displays usage information.

–**m** Specifies that the **Emonitor** daemon should be started. (See /**etc**/**SP**/**Emonitor.cfg** for details.)

Operands

None.

Description

Use this command to start or restart the current system partition based on its switch topology file. (Refer to the **Etopology** command for topology file details.) If the –**m** flag is specified, it will also start the **Emonitor** daemon to monitor nodes on the switch. Refer to the **Emonitor** daemon for additional information. If the **Estart** command is issued when the switch is already running, it causes a switch fault, and messages in flight are lost. Applications using reliable protocols on the switch, such as TCP/IP and the MPI User Space library, recover from switch faults. Applications using unreliable protocols on the switch do not recover from switch faults. For this reason, it is recommended that you should be aware of what applications or protocols you are running before you issue the **Estart** command. Since the **Estart** command uses the SP **rsh** command, proper authentication and authorization to issue this command is necessary.

SP Switch Notes:

If you have an SP Switch installed on your system, an oncoming primary node as selected via **Eprimary** is established as primary during **Estart**. If necessary, the topology file is distributed to partition nodes during **Estart**. The topology file to be used is distributed to each of the nodes in the system partition via the SP Ethernet:

- If an /**etc**/**SP**/**expected.top** file exists on the oncoming primary node
- If the topology file being used is from the System Data Repository (SDR) and not all the nodes in the system partition have the topology file

Otherwise, the topology file is already resident on the nodes and does not need to be distributed.

Files

/etc/SP/expected.top.1nsb_8.0isb.0
The standard topology file for systems with the 8-port High Performance Switch with a maximum of eight nodes.

/etc/SP/expected.top.1nsb.0isb.0
The standard topology file for one Node Switch Board (NSB) system or a maximum of 16 nodes.

/etc/SP/expected.top.2nsb.0isb.0
The standard topology file for two NSB systems or a maximum of 32 nodes.

/etc/SP/expected.top.3nsb.0isb.0
The standard topology file for three NSB systems or a maximum of 48 nodes.

/etc/SP/expected.top.4nsb.0isb.0
The standard topology file for four NSB systems or a maximum of 64 nodes.

/etc/SP/expected.top.5nsb.0isb.0
The standard topology file for five NSB systems or a maximum of 80 nodes.

/etc/SP/expected.top.5nsb.4isb.0
The standard topology file for five NSB and four Intermediate Switch Board (ISB) systems or a maximum of 80 nodes. This is an advantage-type network with a higher bisectional bandwidth.

/etc/SP/expected.top.6nsb.4isb.0
The standard topology file for six NSB and four ISB systems or a maximum of 96 nodes.

/etc/SP/expected.top.7nsb.4isb.0
The standard topology file for seven NSB and four ISB systems or a maximum of 112 nodes.

/etc/SP/expected.top.8nsb.4isb.0
The standard topology file for eight NSB and four ISB systems or a maximum of 128 nodes.

/etc/SP/expected.top.1nsb_8.0isb.1
The standard topology file for systems with an SP Switch-8 and a maximum of eight nodes.

/etc/SP/Emonitor.cfg
The list of nodes that the user wants monitored via the **Emonitor** daemon (not partition sensitive).

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Etopology**, **Eunfence**, **Eunpartition**

Refer to *IBM Parallel System Support Programs for AIX: System Planning* for details about system partition topology files.

Examples

1. To start the High Performance Switch, enter:

   ```
   Estart
   ```

2. To start the High Performance Switch and the **Emonitor** daemon, enter:

   ```
   Estart -m
   ```

Etopology

Purpose

Etopology – Stores or reads a switch topology file into or out of the System Data Repository (SDR).

Syntax

Etopology [–**h**] [–**read**] *switch_topology_file*

Flags

–**h** Displays usage information.

–**read** Retrieves the current switch topology file out of the SDR and stores it in the specified *switch_topology_file*. If –**read** is not specified, the specified *switch_topology_file* will be stored in the SDR.

Operands

switch_topology_file
Specifies the full path name of the file into which the current SDR switch topology is to be copied, or the full path name of a switch topology file to store in the SDR. A sequence number is appended to this file name when it is stored in the SDR. This is used to ensure that the appropriate topology file is distributed to the nodes of the system partition.

Description

Use this command to store or retrieve the *switch_topology_file* into or out of the SDR. The switch topology file is used by switch initialization when starting the switch for the current system partition. It is stored in the SDR and can be overridden by having a switch topology file in the /**etc**/**SP** directory named **expected.top** on the switch primary node.

If you have an SP Switch installed on your system, the current topology file is copied to each node of the subject system partition during an **Estart** and to each targeted node for an **Eunfence**.

Files

/etc/SP/expected.top.1nsb_8.0isb.0
The standard topology file for systems with the 8-port High Performance Switch with a maximum of eight nodes.

/etc/SP/expected.top.1nsb.0isb.0
The standard topology file for one Node Switch Board system or a maximum of 16 nodes.

/etc/SP/expected.top.2nsb.0isb.0
The standard topology file for two NSB systems or a maximum of 32 nodes.

/etc/SP/expected.top.3nsb.0isb.0
The standard topology file for three NSB systems or a maximum of 48 nodes.

/etc/SP/expected.top.4nsb.0isb.0
The standard topology file for four NSB systems or a maximum of 64 nodes.

/etc/SP/expected.top.5nsb.0isb.0
The standard topology file for five NSB systems or a maximum of 80 nodes.

/etc/SP/expected.top.5nsb.4isb.0
The standard topology file for five NSB and four Intermediate Switch Board (ISB) systems or a maximum of 80 nodes. This is an advantage-type network with a higher bisectional bandwidth.

/etc/SP/expected.top.6nsb.4isb.0
The standard topology file for six NSB and four ISB systems or a maximum of 96 nodes.

/etc/SP/expected.top.7nsb.4isb.0
The standard topology file for seven NSB and four ISB systems or a maximum of 112 nodes.

/etc/SP/expected.top.8nsb.4isb.0
The standard topology file for eight NSB and four ISB systems or a maximum of 128 nodes.

/etc/SP/expected.top.1nsb_8.0isb.1
The standard topology file for systems with an SP Switch-8 and a maximum of eight nodes.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Eunfence**, **Eupartition**

Refer to the *IBM Parallel System Support Programs for AIX: System Planning* for information on system partition configurations and topology files.

Examples

1. To store a topology file for a system with up to 96 nodes in the SDR, enter:

   ```
   Etopology /etc/SP/expected.top.6nsb.4isb.0
   ```

2. To store a topology file for a system with up to 16 nodes in the SDR, enter:

   ```
   Etopology /etc/SP/expected.top.1nsb.0isb.0
   ```

3. To retrieve a topology file out of the SDR and store it to a file, enter:

   ```
   Etopology -read /tmp/temporary.top
   ```

Eunfence

Purpose

Eunfence – Adds an SP node to the current *active* switch network that was previously removed from the network.

Syntax

Eunfence [–**h** | [–**G**] *node_specifier* [*node_specifier2*] ...

Flags

–**h** Displays usage information.

–**G** Unfences all valid nodes in the list of nodes regardless of system partition boundaries. If the –**G** flag is not used, the **Eunfence** command will only unfence the nodes in the current system partition. All other specified nodes will not be unfenced and a nonzero return code is returned.

Operands

node_specifier
Specifies a list of nodes that is to rejoin the current switch network. It can be a list of host names, IP addresses, node numbers, frame,slot pairs, or a node group.

Description

Use this command to allow a node to rejoin the current switch network that was previously removed with the **Efence** command.

You can also use this command to allow a node to rejoin the switch network if that node was previously removed from the SP Switch network due to a switch or adapter error.

SP Switch Note:

Eunfence first distributes the current topology file to the nodes before they can be unfenced.

High Performance Switch Note:

The **Eunfence** command cannot unfence a fenced node if a switch fault occurred or if **Estart** ran after the node was fenced. You must do another **Estart** to unfence the node.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunpartition**

Examples

1. To unfence a node by IP address, enter:

   ```
   Eunfence 129.33.34.1
   ```

2. To unfence two nodes by host name, enter:

   ```
   Eunfence r11n01 r11n04
   ```

3. To unfence several nodes by node number, enter:

   ```
   Eunfence 34 43 20 76 40
   ```

4. To unfence node 14 of frame 2 by frame,slots pairs, enter:

   ```
   Eunfence 2,14*
   ```

5. If the current system partition has nodes with node numbers 1, 2, 5, and 6 and another system partition has nodes with node numbers 3, 4, 7, and 8, issuing the command:

   ```
   Eunfence 5 6 7 8
   ```

   unfences nodes 5 and 6, but not nodes 7 and 8. As a result, the command returns a nonzero return code.

6. To successfully unfence the nodes in example 5 with the same system partitions, use the **-G** flag as follows:

   ```
   Eunfence -G 5 6 7 8
   ```

Eunpartition

Purpose

Usage Note

Use this command only if you have an SP Switch installed on your system.

Eunpartition – Prepares a system partition for merging with a neighboring system partition.

Syntax

Eunpartition [–**h**]

Flags

–**h** Displays usage information.

If a flag is not specified, **Eunpartition** examines the SP_NAME shell variable and selects a system partition based on its current setting.

Operands

None.

Description

Use this command to prepare a partitioned configuration for a new system partition definition within an SP cluster.

This command must be executed for each system partition prior to the **spapply_config** command to redefine system partitions. Since this command uses the SP **rsh** command, proper authentication and authorization to issue this command is required.

If you specify **Eunpartition** in error, it will quiesce the primary and primary backup nodes. If this occurs, you must use **Estart** to restart the switch.

Security

You must have root privilege to run this command.

Related Information

Commands: **Eannotator**, **Eclock**, **Eduration**, **Efence**, **Eprimary**, **Equiesce**, **Estart**, **Etopology**, **Eunfence**

Examples

To prepare the current system partition for repartitioning as specified by SP_NAME, enter:

```
Eunpartition
```

export_clients

Purpose

export_clients – Creates or updates the Network File System (NFS) export list for a boot/install server.

Syntax

export_clients [–**h**]

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken.

Operands

None.

Description

Use this command to create or update the NFS export list on a boot/install server node.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/export_clients

Related Information

Commands: **setup_server**

Examples

To create or update the NFS export list on a boot/install server node, enter:

```
export_clients
```

ext_srvtab

Purpose

ext_srvtab – Extracts service key files from the authentication database.

Syntax

ext_srvtab [–**n**] [–**r** *realm*] [*instance ...*]

Flags

–**n** If specified, the master key is obtained from the master key cache file. Otherwise, **ext_srvtab** prompts the user to enter the master key interactively.

–**r** If specified, the realm fields in the extracted file match the given realm rather than the local realm.

Operands

instance Specifies an instance name. On the SP system, service instances consist of the short form of the network names for the hosts on which the service runs.

Description

The **ext_srvtab** command extracts service key files from the authentication database. The master key is used to extract service key values from the database. For each *instance* specified on the command line, the **ext_srvtab** command creates a new service key file in the current working directory with a file name of *instance*-new-srvtab which contains all the entries in the database with an instance field of *instance*. This new file contains all the keys registered for instances of services defined to run on that host. A user must have read access to the authentication database to execute this command. This command can only be issued on the system on which the authentication database resides.

Files

instance-**new-srvtab**
Service key file generated for *instance*.

/var/kerberos/database/principal.pag, /var/kerberos/database/principal.dir
Files containing the authentication database.

/**.k** Master key cache file.

Related Information

Commands: **kadmin**, **ksrvutil**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Examples

If a system has three network interfaces named as follows:

```
ws3e.abc.org
ws3t.abc.org
ws3f.finet.abc.org
```

to re-create the server key file on this workstation (that is an SP authentication server), user root could do the following:

```
# create a new key file in the /tmp  directory for each instance
# Combine the instance files into a single file for the hostname.
# Delete temporary files and protect key file
cd /tmp
/usr/kerberos/etc/ext_srvtab -n ws3e ws3t ws3f
/bin/cat ws3e-new-srvtab ws3t-new-srvtab ws3f-new-srvtab \
   >/etc/krb-srvtab
/bin/rm ws3e-new-srvtab ws3t-new-srvtab ws3f-new-srvtab
/bin/chmod 400 /etc/krb-srvtab
```

ha.vsd

Purpose

ha.vsd – Queries and controls the rvsd subsystem of IBM Recoverable Virtual Shared Disk.

Syntax

ha.vsd {**debug**[*n*] | **query** | **quorum** *n* | **qsrc** | **refresh** | **reset** | **start** | **stop** }

Flags

None.

Operands

debug [*n*] According to the value of **n**, turns debugging on at one of two levels, or turns debugging off. Debugging data is placed in **/var/tmp/admin/SPLOG**. The rvsd subsystem must be in the inactive state when you issue this command.

Use this operand under the direction of your IBM service representative.

n=	Result
1	Turns on level 1 debugging
2	Turns on level 2 debugging
Any other number or blank	Turns debugging off

query Displays the current status of the rvsd subsystem in detail.

quorum *n* Sets the value of the quorum, the number of nodes that must be active to direct recovery. Usually, quorum is defined as a majority of the nodes that are defined as IBM Virtual Shared Disk nodes in a system partition, but this command allows you to override that definition. The rvsd subsystem must be in the active state when you issue this command.

qsrc Displays a short form of the status of the rvsd subsystem.

refresh Sets the value of the quorum back to a majority of the the nodes defined as IBM Virtual Shared Disk nodes in a system partition. This operand reverses the effect of **ha.vsd quorum**. The rvsd subsystem must be in the active state when you issue this command.

reset Stops and restarts the rvsd subsystem.

start Changes the state of the rvsd subsystem to active (from inactive).

stop Changes the state of the rvsd subsystem to inactive (from active).

ha.vsd

Description

Use this command to display information about the rvsd subsystem, to change the number of nodes needed for quorum, and to change the status of the subsystem.

You can start the rvsd subsystem with the VSD Perspective. Type `spvsd` and select actions for IBM VSD nodes.

Files

/var/tmp/admin/SPLOG

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Note: The **query** and **qsrc** subcommands have no exit values.

Security

You must have root privilege to issue the **debug**, **quorum**, **refresh**, **reset**, **start**, and **stop** subcommands.

Implementation Specifics

This command is part of the IBM Recoverable Virtual Shared Disk Licensed Program Product (LPP).

Prerequisite Information

See “Using the IBM Recoverable Virtual Shared Disk Software” in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Location

/usr/lpp/csd/bin/ha.vsd

Related Information

Commands: **hc.vsd**

Examples

1. To stop the rvsd subsystem and restart it, enter:

   ```
   ha.vsd reset
   ```

   The system returns the messages:

   ```
   Waiting for the rvsd subsystem to exit.
   rvsd subsystem exited successfully.
   Starting rvsd subsystem.
   rvsd subsystem started PID=xxx.
   ```

2. To change the quorum to five nodes of a 16-node SP system, enter:

   ```
   ha.vsd quorum 5
   ```

   The system returns the message:

   ```
   Quorum has been changed from 8 to 5.
   ```

hacws_verify

Purpose

hacws_verify – Verifies the configuration of both the primary and backup High Availability Control Workstation (HACWS) control workstations.

Syntax

hacws_verify

Flags

None.

Operands

None.

Description

Use this command to verify that the primary and backup control workstations are properly configured to provide HACWS services to the SP system. The **hacws_verify** command inspects both the primary and backup control workstations to verify the following:

- The HACWS software is properly configured
- The resources required to provide control workstation services have been identified to the HACMP/6000 software
- The HACWS data stored in the System Data Repository (SDR) is correct
- The HACMP event scripts supplied by HACWS have been identified to the HACMP/6000 software

Both the primary and backup control workstations must be running and capable of executing remote commands via the **/usr/lpp/ssp/rcmd/bin/rsh** command.

The system administrator should run the **hacws_verify** command after HACWS is initially configured. After that, the **hacws_verify** command can be run at any time.

Exit Values

0 Indicates that no problems were found with the HACWS configuration.

nonzero Indicates that problems were found with the HACWS configuration.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the HACWS option.

hacws_verify

Location

/usr/sbin/hacws/hacws_verify

Related Information

SP Commands: **install_hacws**, **rsh**, **spcw_addevents**

haemcfg

Purpose

haemcfg – Compiles the Event Management objects in the System Data Repository (SDR) and places the compiled information into a binary Event Management Configuration Database (EMCDB) file

Syntax

haemcfg [–**c**] [–**n**]

Flags

–**c** Indicates that you want to check the data in the System Data Repository (SDR) without building the Event Management Configuration Database (EMCDB).

–**n** Indicates that you want to build a test copy of the EMCDB in the current directory.

Operands

None.

Description

The **haemcfg** utility command builds the Event Management Configuration Database (EMCDB) file for a system partition. If no flags are specified, the **haemcfg** command:

- Compiles the Event Management objects in the System Data Repository (SDR)
- Places the compiled information into a binary Event Management Configuration Database (EMCDB) file in a staging directory as **/spdata/sys1/ha/cfg/em.***syspar_name***.cdb**, where *syspar_name* is the system partition name
- Updates the **haem_cdb_version** attribute in the SDR **Syspar** class for the system partition with the current EMCDB version string. The EMCDB version string contains a timestamp and a sequence number.

To place the new EMCDB into production, you must shut down and restart all of this system partition's Event Manager daemons: the daemon on the control workstation and the daemon on each of the system partition's nodes. When the Event Management daemon restarts, it copies the EMCDB from the staging directory to the production directory. The name of the production EMCDB is **/etc/ha/cfg/em.***syspar_name***.cdb**.

If you want to test a new EMCDB, IBM recommends that you create a separate system partition for that purpose.

You must create a distinct EMCDB file for each system partition on the IBM RS/6000 Scalable POWERparallel Systems (SP). To build an EMCDB file, you must be executing on the control workstation and you must set the **SP_NAME** environment variable to the appropriate system partition name before you issue the command.

Before you build or replace an EMCDB, it is advisable to issue the **haemcfg** command with the debugging flags.

The –**c** flag lets you check the validity of the Event Management data that resides in the SDR. This data was previously loaded through the **haemloadcfg** command. If any of the data is invalid, the command writes an error message that describes the error.

When the –**c** flag is processed, the command validates the data in the SDR, but does not create a new EMCDB file and does not update the EMCDB version string in the SDR.

The –**n** flag lets you build a test EMCDB file in the current directory. If anything goes wrong with the creation of the new file, the command writes an error message that describes the error.

When the –**n** flag is processed, the command uses the data in the SDR to create a test EMCDB file in the current directory, but it does not update the EMCDB version string in the SDR. If any of the data in the SDR is invalid, the command stops at the first error encountered.

If you specify both flags on the command line, the **haemcfg** command performs the actions of the –**c** flag.

After you have checked the data and the build process, issue the **haemcfg** command without any flags. This builds the new EMCDB file, places it in the /**spdata**/**sys1**/**ha**/**cfg** directory, and updates the EMCDB version string in the SDR.

Files

/**spdata**/**sys1**/**ha**/**cfg**/**em.***syspar_name***.cdb**
: Contains the most recently compiled EMCDB file for the system partition specified by *syspar_name*. This file will be placed into production when all of the Event Management daemons in the system partition are next restarted.

/**etc**/**ha**/**cfg**/**em.***syspar_name***.cdb**
: Contains the production EMCDB file for the system partition specified by *syspar_name*. This EMCDB file is currently in use by the Event Management subsystem.

Standard Output

When the command executes successfully, it writes the following informational messages:

```
Reading Event Management data for partition syspar_name
CDB=new_EMCDB_file_name Version=EMCDB_version_string
```

Standard Error

This command writes error messages (as necessary) to standard error.

Errors can result from causes that include:

- Internal space allocation errors
- SDR access errors
- Errors that occur while trying to access system partition data in the SDR
- Invalid Event Management data in the SDR
- EMCDB file access errors
- Insufficient user authorization for the command
- Trying to update the EMCDB in the **/spdata/sys1/ha/cfg** directory from a node of the system partition, rather than from the control workstation.

For a listing of the errors that the **haemcfg** command can produce, see *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred. It is accompanied by one or more error messages that indicate the cause of the error.

Security

To place an EMCDB file for a system partition into the **/spdata/sys1/ha/cfg** directory, you must be running with an effective user ID of **root** on the control workstation. Before running this command, you must set the **SP_NAME** environment variable to the appropriate system partition name.

Restrictions

To place an EMCDB file for a system partition into the **/spdata/sys1/ha/cfg** directory, you must be running with an effective user ID of **root** on the control workstation. Before running this command, you must set the **SP_NAME** environment variable to the appropriate system partition name.

If you run the **haemcfg** command without any flags, the command stops at the first error it encounters. With the **-c** flag on, the command continues, letting you obtain as much debugging information as possible in one pass. To reduce your debugging time, therefore, run the command with the debugging flags first.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For a general overview of configuring Event Management, see "The Event Management Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*

For a description of the SDR classes and attributes that are related to the EMCDB, see *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

Location

/usr/lpp/ssp/bin/haemcfg

Related Information

Commands: **haemloadcfg**

Examples

1. To validate the Event Management data in the System Data Repository (without creating a new EMCDB file), enter:

   ```
   haemcfg -c
   ```

   If there are any errors in the data, the command writes appropriate error messages.

   To fix the errors, replace the data in the SDR. For more information, see the man page for the **haemloadcfg** command.

2. To create a test EMCDB file in the current directory, enter:

   ```
   haemcfg -n
   ```

   If there are any problems in creating the file, the command writes appropriate error messages.

3. To compile a new EMCDB file for a system partition from the Event Management data that resides in the SDR and place it into the staging directory:

   a. Make sure you are executing with root authority on the control workstation.

   b. Make sure that the **SP_NAME** environment variable is set to the name of the appropriate system partition.

   c. Enter:

      ```
      haemcfg
      ```

      In response, the command creates a new EMCDB file, places it in the staging directory as **/spdata/sys1/ha/cfg/em.***syspar_name***.cdb**, where *syspar_name* is the name of the current system partition, and updates the EMCDB version string in the SDR.

haemctrl script

Purpose

haemctrl – A control script that starts the Event Management subsystem.

Syntax

haemctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem.
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystem.
–**o**	Turns tracing off for the subsystem.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

Event Management is a distributed subsystem of PSSP that provides a set of high availability services for the IBM RS/6000 Scalable POWERparallel Systems (SP). By matching information about the state of system resources with information about resource conditions that are of interest to client programs, it creates events. Client programs can use events to detect and recover from system failures, thus enhancing the availability of the SP system.

The **haemctrl** control script controls the operation of the Event Management subsystem. The subsystem is under the control of the System Resource Controller (SRC) and belongs to a subsystem group called **haem**. Associated with each subsystem is a daemon.

An instance of the Event Management subsystem executes on the control workstation and on every node of a system partition. Because Event Management provides its services within the scope of a system partition, its subsystem is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the Event Management subsystem group is organized as follows:

Subsystem Event Management

Subsystem Group	**haem**
SRC Subsystem	**haem**
	The **haem** subsystem is associated with the **haemd** daemon.
	The subsystem name on the nodes is **haem**. There is one of each subsystem per node and it is associated with the system partition to which the node belongs.
	On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **haem.sp_prod** and **haem.sp_test**.
Daemons	**haemd**
	The **haemd** daemon provides the Event Management services.

The **haemctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **haemctrl** script provides a variety of controls for operating the Event Management subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off

Before performing any of these functions, the script obtains the current system partition name and IP address (using the **spget_syspar** command) and the node number (using the **node_number**) command. If the node number is zero, the control script is running on the control workstation.

Except for the clean function, all functions are performed within the scope of the current system partition.

Adding the Subsystem

When the –**a** flag is specified, the control script uses the **mkssys** command to add the Event Management subsystem to the SRC. The control script operates as follows:

1. It makes sure that the **haem** subsystem is stopped.
2. It gets the port number for the **haem** subsystem for this system partition from the **Syspar_ports** class of the System Data Repository (SDR) and ensures that the port number is set in the **/etc/services** file. If there is no port number in the SDR and this script is running on the control workstation, the script obtains a port number. If the script is running on a node and there is no port number in the SDR, the script ends with an error. The range of valid port numbers is 10000 to 10100, inclusive.

   The service name that is entered in the **/etc/services** file is **haem.***syspar_name*.

3. It removes the **haem** subsystem from the SRC (just in case it is still there).
4. It adds the **haem** subsystem to the SRC. On the control workstation, the IP address of the system partition is specified to be supplied as an argument to the daemon by the **mkssys** command.
5. It adds an entry for the **haem** group to the **/etc/inittab** file. The entry ensures that the group is started during boot. However, if **haemctrl** is running on a High Availability Control Workstation (HACWS), no entry is made in the **/etc/inittab** file. Instead, HACWS manages starting and stopping the group.
6. On the control workstation, it creates the Event Management Configuration Database (EMCDB). First, it runs the **haemloadcfg** command to load the SDR with the Event Management configuration data that is contained in the **haemloadlist** file. Then, it runs the **haemcfg** command to compile the data in the SDR and create the binary Event Management Configuration Database. Any errors that occur are written to a log file named **/var/ha/log/em.loadcfg.***syspar_name*.

   For more information about configuring Event Management data, see the *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

   Then it gets the port number for the subsystem from the **SP_ports** class of the System Data Repository (SDR) and ensures that the port number is set in the **/etc/services** file. This port number is used for remote connections to Event Management daemons that are running on the control workstation. If there is no port number in the SDR, the script obtains one and sets it in the **/etc/services** file. The range of valid port numbers is 10000 to 10100, inclusive.

   The service name is **haemd**.

Starting the Subsystem

When the **–s** flag is specified, the control script uses the **startsrc** command to start the Event Management subsystem, **haem**.

Stopping the Subsystem

When the **–k** flag is specified, the control script uses the **stopsrc** command to stop the Event Management subsystem, **haem**.

Deleting the Subsystem

When the **–d** flag is specified, the control script uses the **rmssys** command to remove the Event Management subsystem from the SRC. The control script operates as follows:

1. It makes sure that the **haem** subsystem is stopped.
2. It removes the **haem** subsystem from the SRC using the **rmssys** command.
3. It removes the port number from the **/etc/services** file.
4. If there are no other subsystems remaining in the **haem** group, it removes the entry for the **haem** group from the **/etc/inittab** file.

Cleaning Up the Subsystems

When the –**c** flag is specified, the control script stops and removes the Event Management subsystems for all system partitions from the SRC. The control script operates as follows:

1. It stops all instances of subsystems in the subsystem group in all partitions, using the **stopsrc -g haem** command.
2. It removes the entry for the **haem** group from the /**etc**/**inittab** file.
3. It removes all instances of subsystems in the subsystem group in all partitions from the SRC using the **rmssys** command.
4. It removes all Event Management entries from the /**etc**/**services** file. These include the port numbers for the subsystems as well as the port number used for remote connections.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **haemd** daemon, using the **haemtrcon** command.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off for the **haemd** daemon, using the **haemtrcoff** command.

Refreshing the Subsystem

The –**r** flag has no effect for this subsystem.

Logging

While it is running, the Event Management daemon normally provides information about its operation and errors by writing entries to the AIX error log. If it cannot, errors are written to a log file called /**var**/**ha**/**log**/**em.default.***syspar_name*.

Files

/**var**/**ha**/**log**/**em.default.***syspar_name*
: Contains the default log of the **haemd** daemon on the system partition named *syspar_name*.

/**var**/**ha**/**log**/**em.loadcfg.***syspar_name*
: Contains a log of any errors that occurred while creating the Event Management Configuration Database for the system partition named *syspar_name* using the **haemcfg** command.

/**var**/**ha**/**log**/**em.trace.***syspar_name*
: Contains the trace log of the **haemd** daemon on the system partition named *syspar_name*.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Event Management Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*

IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/haemctrl

Related Information

Commands: **haemcfg**, **haemd**, **haemloadcfg**, **haemtrcoff**, **haemtrcon**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Event Management subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -a
   ```

2. To start the Event Management subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -s
   ```

3. To stop the Event Management subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -k
   ```

4. To delete the Event Management subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -d
   ```

5. To clean up the Event Management subsystem on all system partitions, enter:

   ```
   haemctrl -c
   ```

6. To turn tracing on for the Event Management daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -t
   ```

7. To turn tracing off for the Event Management daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   haemctrl -o
   ```

8. To display the status of all of the subsystems in the Event Management SRC group, enter:

   ```
   lssrc -g haem
   ```

9. To display the status of an individual Event Management subsystem on a node, enter:

   ```
   lssrc -s haem
   ```

   To display the status of an individual Event Management subsystem on the control workstation, enter:

   ```
   lssrc -s haem.syspar_name
   ```

   where *syspar_name* is the system partition name.

10. To display detailed status about an individual Event Management subsystem on a node, enter:

    ```
    lssrc -l -s haem
    ```

    To display detailed status about an individual Event Management subsystem on the control workstation, enter:

    ```
    lssrc -l -s haem.syspar_name
    ```

    where *syspar_name* is the system partition name.

    In response, the system returns information that includes the running status of the subsystem, the settings of trace flags, the version number of the Event Management Configuration Database, the time the subsystem was started, the connection status to Group Services and peer Event Management subsystem, and the connection status to Event Management clients, if any.

11. To display the status of all of the daemons under SRC control, enter:

    ```
    lssrc -a
    ```

haemd daemon

Purpose

haemd – The Event Manager daemon, which observes resource variable instances that are updated by resource monitors and generates and reports events to client programs

Syntax

haemd [*syspar_IPaddr*]

Flags

None.

Operands

syspar_IPaddr
Specifies the IP address of the system partition in which the **haemd** daemon is to execute. If the daemon is executing on the control workstation, this argument must be specified. Otherwise, the argument is ignored, if present.

Description

The **haemd** daemon is the Event Manager daemon. The daemon observes resource variable instances that are updated by resource monitors and generates and reports events to client programs.

One instance of the **haemd** daemon executes on the control workstation for each system partition. An instance of the **haemd** daemon also executes on every node of a system partition. The **haemd** daemon is under System Resource Controller (SRC) control.

Because the daemon is under SRC control, it cannot be started directly from the command line. It is normally started by the **haemctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system. If you must start or stop the daemon directly, use the **haemctrl** command.

For more information about the Event Manager daemon, see the **haemctrl** man page.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Event Management Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*

IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/haemd

Related Information

Commands: **haemctrl**

Examples

See the **haemctrl** command.

haemloadcfg

Purpose

haemloadcfg – Loads Event Management configuration data into the System Data Repository (SDR)

Syntax

haemloadcfg *loadlist_file*

Flags

None.

Operands

loadlist_file The name of the file that contains the Event Management configuration data to be loaded into the SDR. To load the default PSSP configuration data, specify **/usr/lpp/ssp/install/config/haemloadlist**.

Description

The **haemloadcfg** utility command loads Event Management configuration data into the SDR. Note that before you invoke **haemloadcfg**, you must ensure that the **SP_NAME** environment variable is set to the appropriate system partition name.

The configuration data is contained in a load list file, whose format is described by the man page for the **haemloadlist** file. For details on the SDR classes and attributes that you can use to specify Event Management configuration data, see *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

To load the default Event Management configuration data for PSSP, specify the load list file as **/usr/lpp/ssp/install/config/haemloadlist**.

To add Event Management configuration data for other resource monitors, create a file in load list format and specify its name on the command.

The **haemloadcfg** command does not replace existing objects in the SDR. The data in the load list file is matched with the existing objects in the SDR based on key attributes, as follows:

SDR Class	Key Attributes
EM_Resource_Variable	**rvName**
EM_Instance_Vector	**ivResource_name**, **ivElement_name**
EM_Structured_Byte_String	**sbsVariable_name**, **sbsField_name**
EM_Resource_Class	**rcClass**
EM_Resource_Monitor	**rmName**

Note that the way in which the **haemloadcfg** command handles existing SDR objects is different from the way in which the **SDRCreateObjects** command

handles them. The **SDRCreateObjects** command creates a new object as long as the attributes, taken as a group, are unique.

Accordingly, to change a nonkey attribute of an object that already exists in the SDR, you should use the following procedure:

1. Change the object's attribute in the load list file.
2. Use the **SDRDeleteObjects** command to delete the object from the SDR.
3. Run the **haemloadcfg** command, specifying the changed load list file.

In response, the **haemloadcfg** command creates a new object in the SDR for the object that you changed in the load list file. Because the key attributes of all of the other objects in the load list file did not change and the objects already exist, the **haemloadcfg** command does not change any of the other objects in the SDR.

Files

/usr/lpp/ssp/install/config/haemloadlist
Contains the default configuration data for the Event Management subsystem.

Standard Error

This command writes error messages (as necessary) to standard error.

For a listing of the errors that the **haemloadcfg** command can produce, see *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred. It is accompanied by one or more error messages that indicate the cause of the error.

Security

You must have the appropriate authority to write to the SDR. You should be running on the control workstation. Before running this command, you must set the **SP_NAME** environment variable to the appropriate system partition name.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

For a general overview of configuring Event Management, see “The Event Management Subsystem” chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

For details on the System Data Repository classes and attributes for Event Management configuration Database, see *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

Location

/usr/lpp/ssp/bin/haemloadcfg

Related Information

Commands: **haemcfg**, **SDRCreateObjects**, **SDRDeleteObjects**

Files: **haemloadlist**

Also, for a description of the SDR classes for Event Management configuration data, see *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

Examples

1. To load PSSP's default Event Management configuration data into the SDR, enter:

   ```
   haemloadcfg /usr/lpp/ssp/install/config/haemloadlist
   ```

2. To load Event Management configuration data for a new resource monitor that is contained in a file called /**usr**/**local**/**config**/**newrmloadlist**, enter:

   ```
   haemloadcfg /usr/local/config/newrmloadlist
   ```

haemtrcoff

Purpose

haemtrcoff – Turns tracing off for the Event Manager daemon.

Syntax

haemtrcoff –**s** *subsys_name* –**a** *trace_list*

Flags

–**s** *subsys_name*
: Specifies the name of the Event Management subsystem. On a node of a system partition, this is **haem**. On the control workstation, this is **haem.***syspar_name*, where *syspar_name* is the name of the system partition for which you want to specify the subsystem. This argument must be specified.

–**a** *trace_list*
: Specifies a list of trace arguments. Each argument specifies the type of activity for which tracing is to be turned off. At least one argument must be specified. If more than one argument is specified, the arguments must be separated by commas. The list may not include blanks.

Operands

The following trace arguments may be specified:

init Stops tracing the initialization of the Event Manager daemon.

config Stops dumping information from the configuration file.

insts Stops tracing resource variable instances that are handled by the daemon.

rmctrl Stops tracing resource monitor control.

cci Stops tracing the client communication (internal) interface.

emp Stops tracing the event manager protocol.

obsv Stops tracing resource variable observations.

evgn Stops tracing event generation and notification.

reg Stops tracing event registration and unregistration.

pci Stops tracing the peer communication (internal) interface.

msgs Stops tracing all messages that come to and are issued from the daemon.

query Stops tracing queries that are handled by the daemon.

gsi Stops tracing the Group Services (internal) interface.

eval Stops tracing predicate evaluation.

rdi Stops tracing the reliable daemon (internal) interface.

bli Stops tracing the back level (internal) interface, used for handling nodes that are running a level of PSSP that is earlier than PSSP 2.2.

all Stops tracing all activities.

all_but_msgs
Stops tracing all activities except for messages. Message activity is defined by the **msgs** argument.

Description

The **haemtrcoff** command is used to turn tracing off for specified activities of the Event Manager daemon. Trace output is placed in an Event Management trace log for the system partition.

Use this command only under the direction of the IBM Support Center. It provides information for debugging purposes and may degrade the performance of the Event Management subsystem or anything else that is running in the system partition. Do **not** use this command during normal operation.

Files

/var/ha/log/em.trace.*syspar_name*
Contains the trace log of the **haemd** daemon on the system partition named *syspar_name*.

/var/ha/log/em.msgtrace.*syspar_name*
Contains message trace output from the Event Manager daemon on the system partition named *syspar_name*.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Event Management Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/haemtrcoff

Related Information

Commands: **haemctrl**, **haemd**, **haemtrcon**

Examples

In the following examples, the SP system has two system partitions named **sp_prod** and **sp_test**. The instances of the Event Management subsystem on the control workstation of the SP are named **haem.sp_prod** and **haem.sp_test**, respectively. The instance of the Event Management subsystem that runs on any node of either system partition is named **haem**.

1. To turn off all tracing for the Event Management subsystem on the control workstation for the **sp_prod** system partition, login to the control workstation and enter:

   ```
   haemtrcoff -s haem.sp_prod -a all
   ```

2. To turn off all tracing for the Event Management subsystem on one of the nodes of the **sp_test** system partition, login to the node and enter:

   ```
   haemtrcoff -s haem -a all
   ```

3. To turn off all tracing of initialization and configuration for the Event Management subsystem on the control workstation for the **sp_test** system partition, login to the control workstation and enter:

   ```
   haemtrcoff -s haem.sp_test -a init,config
   ```

haemtrcon

Purpose

haemtrcon – Turns tracing on for the Event Manager daemon.

Syntax

haemtrcon –**s** *subsys_name* –**a** *trace_list*

Flags

–**s** *subsys_name*
: Specifies the name of the Event Management subsystem. On a node of a system partition, this is **haem**. On the control workstation, this is **haem.***syspar_name*, where *syspar_name* is the name of the system partition for which you want to specify the subsystem. This argument must be specified.

–**a** *trace_list*
: Specifies a list of trace arguments. Each argument specifies the type of activity for which tracing is to be turned on. At least one argument must be specified. If more than one argument is specified, the arguments must be separated by commas. The list may not include blanks.

Operands

The following trace arguments may be specified:

init Traces the initialization of the Event Manager daemon.

config Dumps information from the configuration file.

insts Traces resource variable instances that are handled by the daemon.

rmctrl Traces resource monitor control.

cci Traces the client communication (internal) interface.

emp Traces the event manager protocol.

obsv Traces resource variable observations.

evgn Traces event generation and notification.

reg Traces event registration and unregistration.

pci Traces the peer communication (internal) interface.

msgs Traces all messages that come to and are issued from the daemon.

query Traces queries that are handled by the daemon.

gsi Traces the Group Services (internal) interface.

eval Traces predicate evaluation.

rdi Traces the reliable daemon (internal) interface.

bli Traces the back level (internal) interface, used for handling nodes that are running a level of PSSP that is earlier than PSSP 2.2.

all Traces all activities.

all_but_msgs
Traces all activities except for messages. Message activity is defined by the **msgs** argument.

Description

The **haemtrcon** command is used to turn tracing on for specified activities of the Event Manager daemon. Trace output is placed in an Event Management trace log for the system partition.

Use this command only under the direction of the IBM Support Center. It provides information for debugging purposes and may degrade the performance of the Event Management subsystem or anything else that is running in the system partition. Do **not** use this command to turn tracing on during normal operation.

Files

/var/ha/log/em.trace.*syspar_name*
Contains the trace log of the **haemd** daemon on the system partition named *syspar_name*.

/var/ha/log/em.msgtrace.*syspar_name*
Contains message trace output from the Event Manager daemon on the system partition named *syspar_name*.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Event Management Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/haemtrcon

Related Information

Commands: **haemctrl**, **haemd**, **haemtrcoff**

Examples

In the following examples, the SP system has two system partitions named **sp_prod** and **sp_test**. The instances of the Event Management subsystem on the control workstation of the SP are named **haem.sp_prod** and **haem.sp_test**, respectively. The instance of the Event Management subsystem that runs on any node of either system partition is named **haem**.

1. To turn on all tracing for the Event Management subsystem on the control workstation for the **sp_prod** system partition, login to the control workstation and enter:

   ```
   haemtrcon -s haem.sp_prod -a all
   ```

2. To turn on all tracing for the Event Management subsystem on one of the nodes of the **sp_test** system partition, login to the node and enter:

```
haemtrcon -s haem -a all
```

3. To turn on all tracing of initialization and configuration for the Event Management subsystem on the control workstation for the **sp_test** system partition, login to the control workstation and enter:

```
haemtrcon -s haem.sp_test -a init,config
```

hagsctrl script

Purpose

hagsctrl – A control script that starts the Group Services subsystems.

Syntax

hagsctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystems.
–**s**	Starts the subsystems.
–**k**	Stops the subsystems.
–**d**	Deletes the subsystems.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystems.
–**o**	Turns tracing off for the subsystems.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

Group Services is a general purpose facility for coordinating and monitoring changes to the state of a client program that is running on a set of nodes on the IBM RS/6000 Scalable POWERparallel Systems (SP). The **hagsctrl** control script controls the operation of the subsystems that are required for Group Services. These subsystems are under the control of the System Resource Controller (SRC) and belong to a subsystem group called **hags**. Associated with each subsystem is a daemon.

An instance of the Group Services subsystem executes on the control workstation and on every node of a system partition. Because Group Services provides its services within the scope of a system partition, its subsystems are said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the Group Services subsystem group is organized as follows:

Subsystem	Group Services
Subsystem Group	**hags**

SRC Subsystems **hags** and **hagsglsm**

The **hags** subsystem is associated with the **hagsd** daemon. The **hagsglsm** subsystem is associated with the **hagsglsmd** daemon.

The subsystem names on the nodes are **hags** and **hagsglsm**. There is one of each subsystem per node and it is associated with the system partition to which the node belongs.

On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **hags.sp_prod**, **hags.sp_test**, **hagsglsm.sp_prod**, and **hagsglsm.sp_test**.

Daemons **hagsd** and **hagsglsmd**

The **hagsd** daemon provides the majority of the Group Services functions.

The **hagsglsmd** daemon provides global synchronization services for the High Performance Switch adapter membership group.

The **hagsctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **hagsctrl** script provides a variety of controls for operating the Group Services subsystems:

- Adding, starting, stopping, and deleting the subsystems
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off

Before performing any of these functions, the script obtains the current system partition name and IP address (using the **spget_syspar** command) and the node number (using the **node_number**) command. If the node number is zero, the control script is running on the control workstation.

Except for the clean function, all functions are performed within the scope of the current system partition.

Adding the Subsystems

When the –**a** flag is specified, the control script uses the **mkssys** command to add the Group Services subsystems to the SRC. The control script operates as follows:

1. It makes sure that both the **hags** and **hagsglsm** subsystems are stopped.
2. It gets the port number for the **hags** subsystem for this system partition from the **Syspar_ports** class of the System Data Repository (SDR) and ensures that the port number is set in the **/etc/services** file. If there is no port number in the SDR and this script is running on the control workstation, the script obtains

a port number. If the script is running on a node and there is no port number in the SDR, the script ends with an error. The range of valid port numbers is 10000 to 10100, inclusive.

The service name that is entered in the **/etc/services** file is **hags.***syspar_name*.

3. It removes the **hags** and **hagsglsm** subsystems from the SRC (just in case they are still there).
4. It adds the **hags** and **hagsglsm** subsystems to the SRC. On the control workstation, the IP address of the system partition is specified to be supplied as an argument to the daemon by the **mkssys** command.
5. It adds an entry for the **hags** group to the **/etc/inittab** file. The entry ensures that the group is started during boot. However, if **hagsctrl** is running on a High Availability Control Workstation (HACWS), no entry is made in the **/etc/inittab** file. Instead, HACWS manages starting and stopping the group.

Starting the Subsystems

When the **-s** flag is specified, the control script uses the **startsrc** command to start the Group Services subsystems, **hags** and **hagsglsm**.

Stopping the Subsystems

When the **-k** flag is specified, the control script uses the **stopsrc** command to stop the Group Services subsystems, **hags** and **hagsglsm**.

Deleting the Subsystems

When the **-d** flag is specified, the control script uses the **rmssys** command to remove the Group Services subsystems from the SRC. The control script operates as follows:

1. It makes sure that both the **hags** and **hagsglsm** subsystems are stopped.
2. It removes the **hags** and **hagsglsm** subsystems from the SRC using the **rmssys** command.
3. It removes the port number from the **/etc/services** file.
4. If there are no other subsystems remaining in the **hags** group, it removes the entry for the **hags** group from the **/etc/inittab** file.

Cleaning Up the Subsystems

When the **-c** flag is specified, the control script stops and removes the Group Services subsystems for all system partitions from the SRC. The control script operates as follows:

1. It stops all instances of subsystems in the subsystem group in all partitions, using the **stopsrc -g hags** command.
2. It removes the entry for the **hags** group from the **/etc/inittab** file.
3. It removes all instances of subsystems in the subsystem group in all partitions from the SRC using the **rmssys** command.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **hagsd** daemon, using the **traceson** command. Tracing is not available for the **hagsglsmd** daemon.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off (returns it to its default level) for the **hagsd** daemon, using the **tracesoff** command. Tracing is not available for the **hagsglsmd** daemon.

Refreshing the Subsystem

The –**r** flag has no effect for this subsystem.

Logging

While they are running, the Group Services daemons provide information about their operation and errors by writing entries in a log file in the /**var**/**ha**/**log** directory.

Each daemon limits the log size to a pre-established number of lines (by default, 10,000 lines). When the limit is reached, the daemon appends the string **.bak** to the name of the current log file and begins a new log. If a **.bak** version already exists, it is removed before the current log is renamed.

Files

/**var**/**ha**/**log**/**hags**_*nodenum_instnum*.*syspar_name*
: Contains the log of the **hagsd** daemons on the nodes.

/**var**/**ha**/**log**/**hags.***syspar_name_nodenum_instnum*.*syspar_name*
: Contains the log of each **hagsd** daemon on the control workstation.

/**var**/**ha**/**log**/**hagsglsm**_*nodenum_instnum*.*syspar_name*
: Contains the log of the **hagsglsmd** daemons on the nodes.

/**var**/**ha**/**log**/**hagsglsm.***syspar_name_nodenum_instnum*.*syspar_name*
: Contains the log of each **hagsglsmd** daemon on the control workstation.

The file names include the following variables:

- *nodenum* is the node number on which the daemon is running
- *instnum* is the instance number of the daemon
- *syspar_name* is the name of the system partition in which the daemon is running.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Group Services Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hagsctrl

Related Information

Commands: **hagsd**, **hagsglsmd**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Group Services subsystems to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hagsctrl -a
   ```

2. To start the Group Services subsystems in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hagsctrl -s
   ```

3. To stop the Group Services subsystems in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hagsctrl -k
   ```

4. To delete the Group Services subsystems from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hagsctrl -d
   ```

5. To clean up the Group Services subsystems on all system partitions, enter:

   ```
   hagsctrl -c
   ```

6. To turn tracing on for the Group Services daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter: enter:

```
hagsctrl -t
```

7. To turn tracing off for the Group Services daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter: enter:

   ```
   hagsctrl -o
   ```

8. To display the status of all of the subsystems in the Group Services SRC group, enter:

   ```
   lssrc -g hags
   ```

9. To display the status of an individual Group Services subsystem, enter:

   ```
   lssrc -s subsystem_name
   ```

10. To display detailed status about an individual Group Services subsystem, enter:

    ```
    lssrc -l -s subsystem_name
    ```

    In response, the system returns information that includes the running status of the subsystem, the number and identity of connected GS clients, information about the Group Services domain, and the number of providers and subscribers in established groups.

11. To display the status of all of the daemons under SRC control, enter:

    ```
    lssrc -a
    ```

hagsd daemon

Purpose

hagsd – A Group Services daemon that provides a general purpose facility for coordinating and monitoring changes to the state of an application that is running on a set of nodes.

Syntax

hagsd [–**p** *syspar_IPaddr*]

Flags

–**p** *syspar_IPaddr*
Specifies the IP address of the system partition in which the **hagsd** daemon is to execute. If the daemon is executing on the control workstation, this argument must be specified. Otherwise, the argument is ignored, if present.

Operands

None.

Description

The **hagsd** daemon is part of the Group Services subsystem, which provides a general purpose facility for coordinating and monitoring changes to the state of an application that is running on a set of nodes. This daemon provides most of the services of the subsystem.

One instance of the **hagsd** daemon executes on the control workstation for each system partition. An instance of the **hagsd** daemon also executes on every node of a system partition. The **hagsd** daemon is under System Resource Controller (SRC) control.

Because the daemon is under SRC control, it is better not to start it directly from the command line. It is normally called by the **hagsctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system. If you must start or stop the daemon directly, use the **startsrc** or **stopsrc** command.

For more information about the Group Services daemons, see the **hagsctrl** man page.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Group Services Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hagsd

Related Information

Commands: **hagsctrl**, **hagsglsmd**

Examples

See the **hagsctrl** command.

hagsglsmd daemon

Purpose

hagsglsmd – A Group Services daemon that provides global synchronization services for the High Performance Switch adapter membership group.

Syntax

hagsglsmd [–**p** *syspar_IPaddr*]

Flags

–**p** *syspar_IPaddr*
: Specifies the IP address of the system partition in which the **hagsglsmd** daemon is to execute. If the daemon is executing on the control workstation, this argument must be specified. Otherwise, the argument is ignored, if present.

Operands

None.

Description

The **hagsglsmd** daemon is part of the Group Services subsystem, which provides a general purpose facility for coordinating and monitoring changes to the state of an application that is running on a set of nodes. This daemon provides global synchronization services for the High Performance Switch adapter membership group.

One instance of the **hagsglsmd** daemon executes on the control workstation for each system partition. An instance of the **hagsglsmd** daemon also executes on every node of a system partition. The **hagsglsmd** daemon is under System Resource Controller (SRC) control.

Because the daemon is under SRC control, it is better not to start it directly from the command line. It is normally called by the **hagsctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system. If you must start or stop the daemon directly, use the **startsrc** or **stopsrc** command.

For more information about the Group Services daemons, see the **hagsctrl** man page.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

"The Group Services Subsystem" chapter of *IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference*.

IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hagsglsmd

Related Information

Commands: **hagsctrl**, **hagsd**

Examples

See the **hagsctrl** command.

hardmon daemon

Purpose

hardmon – Monitors and controls the state of the SP hardware.

Syntax

hardmon [–**B**] [–**r** *poll_rate*] [–**d** *debug_flag*] ...

Flags

–**B** Executes the daemon in diagnostic mode.

–**r** *poll_rate* Specifies the rate, in seconds, at which the daemon polls each frame for state information.

–**d** *debug_flag*
Specifies the daemon debug flag to be set in the daemon. Refer to the **hmadm** command for possible values of *debug_flag*. Multiple –**d** debug flags can be specified.

Operands

None.

Description

hardmon is the Hardware Monitor daemon. The daemon monitors and controls the state of the SP hardware contained in one or more SP frames. This command is not normally executed from the command line. Access to the Hardware Monitor is provided by the **hmmon**, **hmcmds**, **spmon**, **s1term**, and **nodecond** commands. Control of the Hardware Monitor daemon is provided by the **hmadm** command. These commands are the Hardware Monitor “client” commands.

The Hardware Monitor daemon executes on the Monitor and Control Node (MACN). The MACN is that IBM RS/6000 workstation to which the RS-232 lines are connected to the frames. The MACN is one and the same as the control workstation. The daemon is managed by the System Resource Controller (SRC). When the MACN is booted, an entry in /**etc/inittab** invokes the **startsrc** command to start the daemon. The daemon is configured in the SRC to be restarted automatically if it terminates for any reason other than the **stopsrc** command. The SRC subsystem name for the Hardware Monitor daemon is **hardmon**.

hardmon obtains configuration information from the System Data Repository (SDR). The SP_ports object class specifies the port number that the daemon is to use to accept TCP/IP connections from the client commands. The port number is obtained from the object whose *daemon* attribute value matches **hardmon** and whose *host_name* attribute value matches the host name of the workstation on which the daemon is executing. There must be one **hardmon** object in SP_ports for the MACN. The Frame object class contains an object for each frame in the SP system.

The attributes of interest to the daemon are **frame_number**, **tty**, and **MACN**. When started, the daemon fetches all those objects in the Frame class whose *MACN* attribute value matches the host name of the workstation on which the

daemon is executing. For each frame discovered in this manner, the daemon saves the frame number and opens the corresponding tty device. When all frames have been configured, the daemon begins to poll the frames for state information. Current state and changed state can then be obtained using the **hmmon** and **spmon** commands. The **hmcmds** and **spmon** commands can be used to control the hardware within the frames.

The daemon also reads the file **/spdata/sys1/spmon/hmthresholds** for values used to check boundary conditions for certain state variables. This file should only be changed on request from IBM support. Finally, the **/spdata/sys1/spmon/hmacls** file is read for Access Control List (ACL) information. Refer to the **hmadm** command and the **/spdata/sys1/spmon/hmacls** file for more information on ACLs.

All errors detected by the Hardware Monitor daemon are written to the AIX error log.

The flags in the SRC subsystem object for the **hardmon** subsystem should not normally be changed. For example, if the poll rate is more than 5 seconds, the **nodecond** command can fail with unpredictable results. Upon request from IBM support for more information to aid in problem determination, debug flags can be set using the **hmadm** command.

If the High Availability Control Workstation (HACWS) Frame Supervisor (type 20) or the SEPBU HACWS Frame Supervisor (type 22) is installed in the SP frames, the **-B** flag is used to run the Hardware Monitor daemon in diagnostic mode. This diagnostic mode is used to validate that the frame ID written into the Supervisor matches the frame ID configured in the SDR for that frame. Normally, the frame ID is automatically written into the Supervisor during system installation. The frame ID is written into the frame to detect cabling problems in an HACWS configuration. In a non-HACWS SP configuration, the **-B** flag is useful whenever the RS232 cables between the frames and MACN are changed (but only if one or more frames contain a type 20 or type 22 supervisor). The **hardmon** command can be executed directly from the command line with the **-B** flag, but only after the currently running daemon is stopped using the **stopsrc** command. Diagnostic messages are written to the AIX error log. The daemon exits when all frames are validated.

Frame ID validation is also performed every time the daemon is started by the System Resource Controller. Any frame that has a frame ID mismatch can be monitored, but any control commands to the frame are ignored until the condition is corrected. A frame with a mismatch is noted in the System Monitor Graphical User Interface as well as in the AIX error log. The **hmcmds** command can be used to set the currently configured frame ID into a type 20 or type 22 supervisor after it is verified that the frame is correctly connected to the MACN.

Additional Configuration Information: The Hardware Monitor subsystem also obtains information from the system partition and the Syspar_map object classes in the SDR. While this information is not used by the **hardmon** daemon itself, it is used by the **hardmon** client commands listed under Related Information. Each of these commands executes in the environment of one system partition. If the SP system is not partitioned, these commands execute in the environment of the entire system. In any case, the Syspar_map object class is used to determine which

nodes are contained in the current environment. The attributes of interest are *syspar_name* and *node_number*.

Starting and Stopping the hardmon Daemon

The **hardmon** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **hardmon** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **hardmon**. In order to start the **hardmon** daemon, use the **startsrc** –**s hardmon** command. This starts the daemon with the default arguments and SRC options. The **hardmon** daemon is setup to be respawnable and be the only instance of the **hardmon** daemon running on a particular node or control workstation. Do **not** start the **hardmon** daemon from the command line without using the **startsrc** command to start it.

To stop the **hardmon** daemon, use the **stopsrc** –**s hardmon** command. This stops the daemon and does not allow it to respawn.

To display the status of the **hardmon** daemon, use the **lssrc** –**s hardmon** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4 Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget** –**q "subsysname=hardmon" SRCsubsys** command.

Files

/usr/lpp/ssp/bin/hardmon
Contains the **hardmon** command.

/spdata/sys1/spmon/hmthresholds
Contains boundary values.

/spdata/sys1/spmon/hmacls
Contains Access Control Lists.

Related Information

Commands: **hmadm**, **hmcmds**, **hmmon**, **nodecond**, **spmon**, **s1term**

File: **/spdata/sys1/spmon/hmacls**

Examples

1. To start the **hardmon** daemon, enter:

   ```
   startsrc -s hardmon
   ```

2. To stop the **hardmon** daemon, enter:

   ```
   stopsrc -s hardmon
   ```

3. To display the status of the **hardmon** daemon, enter:

   ```
   lssrc -s hardmon
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **hardmon** daemon, enter:

   ```
   odmget -q "subsysname=hardmon" SRCsubsys
   ```

hats script

Purpose

hats – Starts or restarts Topology Services on a node or on the control workstation.

Syntax

hats

Flags

None.

Operands

None.

Description

Use this command to start the operation of Topology Services for a system partition (the **hatsd** daemon) on the control workstation or on a node within a system partition.

The **hats** script is not normally executed from the command line. It is normally called by the **hatsctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The Topology Services subsystem provides internal services to PSSP components.

Note that the **hats** script issues the **no -o nonlocsrcroute=1** command, which enables IP source routing. Do **not** change this setting, because the Topology Services subsystem requires this setting to work properly. If you change the setting, the Topology Services subsystem and a number of other subsystems that depend on it will no longer operate properly.

The **hatsd** daemon is initially started on the control workstation with the System Resource Controller (SRC), regardless of the level of the system partition. It is respawned automatically if the **hatsd** daemon fails. The SP_NAME environment variable causes selection of the correct topology configuration.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

The “Starting Up and Shutting Down the SP System” chapter and “The System Data Repository” appendix in *IBM Parallel System Support Programs for AIX: Administration Guide*.

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hats

Related Information

Commands: **hatsctrl**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

See the **hatsctrl** command.

hatsctrl script

Purpose

hatsctrl – A control script that starts the Topology Services subsystem.

Syntax

hatsctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem.
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystem.
–**o**	Turns tracing off for the subsystem.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

Topology Services is a distributed subsystem of PSSP that provides information to other PSSP subsystems about the state of the nodes and adapters on the IBM RS/6000 Scalable POWERparallel Systems (SP).

The **hatsctrl** control script controls the operation of the Topology Services subsystem. The subsystem is under the control of the System Resource Controller (SRC) and belongs to a subsystem group called **hats**. Associated with each subsystem is a daemon and a script that configures and starts the daemon.

An instance of the Topology Services subsystem executes on the control workstation and on every node of a system partition. Because Topology Services provides its services within the scope of a system partition, its subsystem is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the Topology Services subsystem group is organized as follows:

Subsystem Topology Services

Subsystem Group **hats**

SRC Subsystem **hats**

The **hats** subsystem is associated with the **hatsd** daemon and the **hats** script. The **hats** script configures and starts the **hatsd** daemon.

The subsystem name on the nodes is **hats**. There is one of each subsystem per node and it is associated with the system partition to which the node belongs.

On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **hats.sp_prod** and **hats.sp_test**.

Daemons **hatsd**

The **hatsd** daemon provides the Topology Services. The **hats** script configures and starts the **hatsd** daemon.

The **hatsctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **hatsctrl** script provides a variety of controls for operating the Topology Services subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off
- Refreshing the subsystem

Before performing any of these functions, the script obtains the current system partition name and IP address (using the **spget_syspar** command) and the node number (using the **node_number**) command. If the node number is zero, the control script is running on the control workstation.

Except for the clean function, all functions are performed within the scope of the current system partition.

Adding the Subsystem

When the –**a** flag is specified, the control script uses the **mkssys** command to add the Topology Services subsystem to the SRC. The control script operates as follows:

1. It makes sure that the **hats** subsystem is stopped.
2. It gets the port number for the **hats** subsystem for this system partition from the **Syspar_ports** class of the System Data Repository (SDR) and ensures that the port number is set in the /**etc**/**services** file. If there is no port number in the SDR and this script is running on the control workstation, the script obtains a port number. If the script is running on a node and there is no port number in the SDR, the script ends with an error. The range of valid port numbers is 10000 to 10100, inclusive.

The service name that is entered in the /**etc**/**services** file is **hats.***syspar_name*.

3. It checks to see if the subsystem is already configured in the SDR. If not, it creates an instance of the **TS_Config** class for this subsystem with default values. The default values are:
   - Heartbeats are sent out a rate of 1 per second (**Frequency** attribute = 1)
   - The number of heartbeats from the neighboring node that can be missed before the neighbor is declared inoperative is 4 (**Sensitivity** attribute = 4)
   - The execution priority is fixed (**Run_FixPri** attribute = 1)
   - The value of the execution priority used on the **set_priority** system call is 38 (**FixPri_Value** attribute = 38).
4. It removes the **hats** subsystem from the SRC (just in case it is still there).
5. It adds the **hats** subsystem to the SRC. On the control workstation, the IP address of the system partition is specified to be supplied as an argument to the daemon by the **mkssys** command.
6. It adds an entry for the **hats** group to the /**etc**/**inittab** file. The entry ensures that the group is started during boot. However, if **hatsctrl** is running on a High Availability Control Workstation (HACWS), no entry is made in the /**etc**/**inittab** file. Instead, HACWS manages starting and stopping the group.

Starting the Subsystem

When the −**s** flag is specified, the control script uses the **startsrc** command to start the Topology Services subsystem, **hats**.

Stopping the Subsystem

When the −**k** flag is specified, the control script uses the **stopsrc** command to stop the Topology Services subsystem, **hats**.

Deleting the Subsystem

When the −**d** flag is specified, the control script uses the **rmssys** command to remove the Topology Services subsystem from the SRC. The control script operates as follows:

1. It makes sure that the **hats** subsystem is stopped.
2. It removes the **hats** subsystem from the SRC using the **rmssys** command.
3. It removes the port number from the /**etc**/**services** file.
4. If there are no other subsystems remaining in the **hats** group, it removes the entry for the **hats** group from the /**etc**/**inittab** file.

Cleaning Up the Subsystems

When the −**c** flag is specified, the control script stops and removes the Topology Services subsystems for all system partitions from the SRC. The control script operates as follows:

1. It stops all instances of subsystems in the subsystem group in all partitions, using the **stopsrc -g hats** command.
2. It removes the entry for the **hats** group from the /**etc**/**inittab** file.
3. It removes all instances of subsystems in the subsystem group in all partitions from the SRC using the **rmssys** command.

4. It removes all entries for the **hats** subsystems from the /**etc**/**services** file.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **hatsd** daemon, using the **traceson** command.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off (returns it to its default level) for the **hatsd** daemon, using the **tracesoff** command.

Refreshing the Subsystem

When the –**r** flag is specified, the control script refreshes the subsystem, using the **hats refresh** command and the **refresh** command.

It rebuilds the information about the node and adapter configuration in the SDR and signals the daemon to read the rebuilt information.

Logging

While it is running, the Topology Services daemon provides information about its operation and errors by writing entries in a log file. The **hatsd** daemon in the system partition named *syspar_name* uses a log file called /**var**/**ha**/**log**/**hats.***syspar_name*.

Files

/**var**/**ha**/**log**/**hats.***syspar_name***.**
Contains the log of the **hatsd** daemon on the system partition named *syspar_name*.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hatsctrl

Related Information

Commands: **hats**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Topology Services subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -a
   ```

2. To start the Topology Services subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -s
   ```

3. To stop the Topology Services subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -k
   ```

4. To delete the Topology Services subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -d
   ```

5. To clean up the Topology Services subsystem on all system partitions, enter:

   ```
   hatsctrl -c
   ```

6. To turn tracing on for the Topology Services daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -t
   ```

7. To turn tracing off for the Topology Services daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hatsctrl -o
   ```

8. To display the status of all of the subsystems in the Topology Services SRC group, enter:

   ```
   lssrc -g hats
   ```

9. To display the status of an individual Topology Services subsystem, enter:

   ```
   lssrc -s subsystem_name
   ```

10. To display detailed status about an individual Topology Services subsystem, enter:

    ```
    lssrc -l -s subsystem_name
    ```

    In response, the system returns information that includes the running status of the subsystem, the number of defined and active nodes, the required number

of active nodes for a quorum, the status of the group of nodes, and the IP addresses of the source node, the group leader, and the control workstation.

11. To display the status of all of the daemons under SRC control, enter:

```
lssrc -a
```

hb script

Purpose

hb – Starts or restarts heartbeat services on a node or on the control workstation.

Syntax

hb [–**spname** *syspar_name*] [–**splevel** *pssp_level*]

{ [**start** | **resume**] | [**stop** | **quiesce**] | **reset** |

[**query** | **qall** | **qsrc**] | **refresh** | **mksrc** *optional_flags* | **rmsrc restore** |

[**debug** | **debug off**] | [**trace on** | **trace off**] }

Flags

–**spname** *syspar_name*
: Executes the command for the system partition specified by the *syspar_name* operand. If this flag is not specified, the name of the system partition given by the value of the **SP_NAME** variable is used.

–**splevel** *pssp_level*
: Sets the system partition level to the value specified by the *pssp_level* operand. Valid levels are: PSSP-1.2, PSSP-2.1, or PSSP-2.2. The default level is PSSP-2.2.

Operands

start | **resume**
: Resumes normal heartbeat services after they have been temporarily suspended with **quiesce** or **stop**.

stop | **quiesce**
: Stops heartbeat services (**hbd**).

reset
: Stops and restarts the heartbeat server on a node. Use this parameter:

  1. After changing relevant node information in the System Data Repository (SDR). (Reset all nodes on the affected system partition and that partition's **hbd** on the control workstation.)
  2. When host_responds consistently does not agree with the state of the node and automatic recovery has not taken place.

query
: Queries the daemon for a partition. The response to a **query** includes heartbeat-specific information.

qall
: Performs the **query** function for each defined partition.

qsrc
: Displays a subsystem definition for a partition.

refresh
: Uses the refresh command to request a daemon refresh.

mksrc *optional_flags*
: Uses the **mkssys** command to create an SRC subsystem object. Additional flags for the command may be specified.

rmsrc Uses the **rmssys** command to remove an SRC subsystem object.

restore Synchronizes the running daemons with the information in the System Data Repository (SDR). This operand removes all entries for the subsystem, creates new entries based on information in the SDR, and starts the subsystems.

[debug | debug off]
Turns debugging on or off.

[trace on | trace off]
Turns additional tracing on or off.

Description

Use this command to control the operation of heartbeat services for a system partition (the **hbd** daemon) on the control workstation or on a node within a system partition.

The **hb** script is not normally executed from the command line. It is normally called by the **hbctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The heartbeat server provides input to the host_responds function within a system partition for the System Monitor through the System Monitor **hr** daemons. It also provides input to the IBM Recoverable Virtual Shared Disk daemons, if that product is installed on the nodes. This involves the following daemons:

hbd The internal heartbeat server on the nodes and the control workstation.

hrd The host responds daemon on the control workstation.

Note: The **hrd** daemon is controlled by the **hr** script. The **hbd** daemon is controlled with this script.

The **hbd** daemon is initially started on the control workstation with the System Resource Controller (SRC), regardless of the level of the system partition. It is respawned automatically if the **hbd** daemon fails. The SP_NAME environment variable causes selection of the correct heartbeat daemon.

The **hbd** daemons communicate with their counterparts on other nodes over the SP Ethernet. The udp **heartbeat** entry in **/etc/services** on all nodes must specify the same port number.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

The “Starting Up and Shutting Down the SP System” chapter and “The System Data Repository” appendix in *IBM Parallel System Support Programs for AIX: Administration Guide*.

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

hb script

Location

/usr/lpp/ssp/bin/hb

Related Information

Commands: **hbctrl**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

See the **hbctrl** command.

hbctrl script

Purpose

hbctrl – A control script that starts the Heartbeat subsystem.

Syntax

hbctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem.
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystem.
–**o**	Turns tracing off for the subsystem.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

The Heartbeat subsystem communicates with several PSSP subsystems as part of providing information about the state of the nodes and adapters on the IBM RS/6000 Scalable POWERparallel Systems (SP).

The **hbctrl** control script controls the operation of the Heartbeat subsystem. The subsystem is under the control of the System Resource Controller (SRC) and belongs to a subsystem group called **hb**. Associated with each subsystem is a daemon and a script that configures and starts the daemon.

An instance of the Heartbeat subsystem executes on the control workstation and on every node of a system partition. Because Heartbeat provides its services within the scope of a system partition, its subsystem is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the Heartbeat subsystem group is organized as follows:

Subsystem Heartbeat

Subsystem Group **hb**

SRC Subsystem **hb**

The **hb** subsystem is associated with the **hbd** daemon and the **hb** script. The **hb** script configures and starts the **hbd** daemon.

The subsystem name on the nodes is **hb**. There is one of each subsystem per node and it is associated with the system partition to which the node belongs.

On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **hb.sp_prod** and **hb.sp_test**.

Daemons **hbd**

The **hbd** daemon provides the Heartbeat services. The **hb** script configures and starts the **hbd** daemon.

The **hbctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **hbctrl** script provides a variety of controls for operating the Heartbeat subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off
- Refreshing the subsystem.

Before performing any of these functions, the script obtains the current system partition name and IP address (using the **spget_syspar** command) and the node number (using the **node_number**) command. If the node number is zero, the control script is running on the control workstation.

Except for the clean function, all functions are performed within the scope of the current system partition.

Adding the Subsystem

When the **–a** flag is specified, the control script uses the **mkssys** command to add the Heartbeat subsystem to the SRC. The control script operates as follows:

1. It makes sure that the **hb** subsystem is stopped.
2. It gets the port number for the **hb** subsystem for this system partition from the **SP_ports** class of the System Data Repository (SDR) and ensures that the port number is set in the **/etc/services** file. If there is no port number in the SDR and this script is running on the control workstation, the script obtains a port number. If the script is running on a node and there is no port number in the SDR, the script ends with an error. The Heartbeat subsystem uses port number 4893.

   The service name that is entered in the **/etc/services** file is **heartbeat**.

3. If this script is running on the control workstation, it checks to see if the subsystem is already configured in the SDR. If not, it creates an instance of the **HB_Config** class for this subsystem with default values. The default values are:
   - Heartbeats are sent out a rate of 1 per second (**Frequency** attribute = 1)
   - The number of heartbeats from the neighboring node that can be missed before the neighbor is declared inoperative is 4 (**Sensitivity** attribute = 4)
   - The execution priority is fixed (**Run_FixPri** attribute = 1)
   - The value of the execution priority used on the **set_priority** system call is 32 (**FixPri_Value** attribute = 32).
4. It invokes the **hb** script with the **mksrc** parameter to add the subsystem to the SRC. On the control workstation, the name of the system partition is also specified on the **hb** script.
5. It adds an entry for the **hb** group to the /**etc**/**inittab** file. The entry ensures that the group is started during boot. However, if **hbctrl** is running on a High Availability Control Workstation (HACWS), no entry is made in the /**etc**/**inittab** file. Instead, HACWS manages starting and stopping the group.

Starting the Subsystem

When the –**s** flag is specified, the control script uses the **hb** command to start the Heartbeat subsystem, **hb**.

Stopping the Subsystem

When the –**k** flag is specified, the control script uses the **hb** command to stop the Heartbeat subsystem, **hb**.

Deleting the Subsystem

When the –**d** flag is specified, the control script uses the **rmssys** command to remove the Heartbeat subsystem from the SRC. The control script operates as follows:

1. It removes the **hb** subsystem from the SRC using the **hb** script with the **rmsrc** parameter.
2. It removes the port number from the /**etc**/**services** file.
3. If there are no other subsystems remaining in the **hb** group, it removes the entry for the **hb** group from the /**etc**/**inittab** file.

Cleaning Up the Subsystems

When the –**c** flag is specified, the control script stops and removes the Heartbeat subsystems for all system partitions from the SRC. The control script operates as follows:

1. It stops all instances of subsystems in the subsystem group in all partitions, using the **stopsrc -g hb** command.
2. It removes the entry for the **hb** group from the /**etc**/**inittab** file.
3. It removes all instances of subsystems in the subsystem group in all partitions from the SRC using the **rmssys** command.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **hbd** daemon, using the **traceson** command.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off (returns it to its default level) for the **hbd** daemon, using the **tracesoff** command.

Refreshing the Subsystem

When the –**r** flag is specified, the control script refreshes the subsystem, using the **hb refresh** command.

Logging

While it is running, the Heartbeat daemon provides information about its operation and errors by writing entries in a log file. The **hbd** daemon in the system partition named *syspar_name* uses a log file called /**var**/**ha**/**log**/**hb.***syspar_name*.

Files

/**var**/**ha**/**log**/**hb.***syspar_name***.**
: Contains the log of the **hbd** daemon on the system partition named *syspar_name*.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/**usr**/**lpp**/**ssp**/**bin**/**hbctrl**

Related Information

Commands: **hb**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Heartbeat subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -a
   ```

2. To start the Heartbeat subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -s
   ```

3. To stop the Heartbeat subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -k
   ```

4. To delete the Heartbeat subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -d
   ```

5. To clean up the Heartbeat subsystem on all system partitions, enter:

   ```
   hbctrl -c
   ```

6. To turn tracing on for the Heartbeat daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -t
   ```

7. To turn tracing off for the Heartbeat daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hbctrl -o
   ```

8. To display the status of all of the subsystems in the Heartbeat SRC group, enter:

   ```
   lssrc -g hb
   ```

9. To display the status of an individual Heartbeat subsystem, enter:

   ```
   lssrc -s subsystem_name
   ```

10. To display detailed status about an individual Heartbeat subsystem, enter:

    ```
    lssrc -l -s subsystem_name
    ```

    In response, the system returns information that includes the running status of the subsystem, the number of defined and active nodes, the required number of active nodes for a quorum, the status of the group of nodes, the frequency and sensitivity values in use for the subsystem, and the IP addresses of the source node, the group leader, and the control workstation.

11. To display the status of all of the daemons under SRC control, enter:

    ```
    lssrc -a
    ```

hc.vsd

Purpose

hc.vsd – Queries and controls the hc subsystem of IBM Recoverable Virtual Shared Disk.

Syntax

hc.vsd {**debug**[*n*] | **mksrc** | **query** | **qsrc** | **reset** | **start** | **stop** }

Flags

None.

Operands

debug [*n*] According to the value of **n**, turns debugging on at one of two levels, or turns debugging off. Debugging data is placed in **/var/tmp/admin/SPLOG**. The hc subsystem must be in the inactive state when you issue this command.

Use this operand under the direction of your IBM service representative.

n=	Result
1	Turns on level 1 debugging
2	Turns on level 2 debugging
Any other number or blank	Turns debugging off

query Displays the current status of the hc subsystem in detail.

qsrc Displays a short form of the status of the hc subsystem.

reset Stops and restarts the hc subsystem.

start Changes the state of the hc subsystem to active (from inactive).

stop Changes the state of the hc subsystem to inactive (from active).

Description

Use this command to display information about the hc subsystem and to change the status of the subsystem.

You can restart the hc subsystem with the VSD Perspective. Type `spvsd` and select actions for IBM VSD nodes.

Files

/var/tmp/admin/SPLOG

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Note: The **query** and **qsrc** subcommands have no exit values.

Security

You must have root privilege to issue the **debug**, **mksrc**, **reset**, **start**, and **stop** commands.

Implementation Specifics

This command is part of the IBM Recoverable Virtual Shared Disk Licensed Program Product (LPP).

Prerequisite Information

See “Using the IBM Recoverable Virtual Shared Disk Software” in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Location

/usr/lpp/csd/bin/hc.vsd

Related Information

Commands: **ha.vsd**

Examples

1. To stop the hc subsystem and restart it, enter:

   ```
   hc.vsd reset
   ```

   The system returns the messages:

   ```
   Waiting for the hc subsystem to exit.
   hc subsystem exited successfully.
   Starting hc subsystem.
   hc subsystem started PID=xxx.
   ```

hmadm

Purpose

hmadm – Administers the Hardware Monitor daemon.

Syntax

hmadm [{–**d** *debug_flag*} ...] *operation*

Flags

–**d** *debug_flag*
Specifies the daemon debug flag to be set or unset in the daemon.

Operands

operation Specifies the administrative action to perform.

The *operation* must be one of the following:

cleard Unsets the daemon debug flag specified by the –**d** flag in the daemon. Multiple –**d** flags can be specified. If no –**d** flags are specified, the **all** debug flag is assumed.

clog Changes the daemon log file. If the log file is growing large, this operation is used to cause the daemon to write to a new log file.

quit Causes the daemon to exit.

setacls Reads the Hardware Monitor access control list configuration file to update the daemon's internal ACL tables. Any Hardware Monitor application or command executing under the ID of a user who has changed or deleted ACLs has its client connection terminated by the daemon. Such applications and commands must be restarted, if possible. ACLs for new users can be added without any effect on executing applications and commands.

This operation must by invoked by the administrator after the administrator modifies the ACL configuration file.

setd Sets the daemon debug flag specified by the –**d** flag in the daemon. Multiple –**d** flags can be specified. If no –**d** flags are specified, the **all** debug flag is assumed.

Description

The **hmadm** command is used to administer the Hardware Monitor daemon. The Hardware Monitor daemon executes on the control workstation and is used to monitor and control the SP hardware. Five administrative actions are supported, as specified by the *operation* operand.

Normally when the daemon exits, it is automatically restarted by the system. If frame configuration information is changed, the **quit** operation can be used to update the system.

The daemon writes debug information and certain error information to its log file. The log file is located in **/var/adm/SPlogs/spmon** and its name is of the form **hmlogfile.***nnn*, where *nnn* is the Julian date of the day the log file was opened by the daemon. The **clog** operation causes the daemon to close its current log file and create a new one using the name **hmlogfile***nnn*, where *nnn* is the current Julian date. If this name already exists, a name of the form **hmlogfile.***nnn_m* is used, where *m* is a number picked to create a unique file name.

There are 15 debug flags supported by the daemon:

all Sets/unsets all of the following flags.

acls Logs the Access Control Lists.

cmdq Logs the contents of the internal queue of commands sent to the frames.

cntrs Logs the daemon internal counters.

dcmds Logs commands sent to the daemon.

fcmds Logs commands sent to the frames.

ipl Logs interested party lists.

pckts Logs packets received from the frames in **/var/adm/SPlogs/spmon/hm_frame_packet_dump**.

polla Logs poll list array.

rsps Logs responses sent to clients in **/var/adm/SPlogs/spmon/hm_response_dump**.

socb Logs client socket session information.

s1data Logs data sent to S1 serial ports in **/var/adm/SPlogs/spmon/hm_s1data_dump**.

s1refs Logs S1 serial port reference counts and connections.

ttycb Logs ttycb control blocks.

tvars Logs boundary values used in checking temperatures, amperages, and volts.

This command uses the SP Hardware Monitor. Therefore, the user must be authorized to access the Hardware Monitor subsystem and must have administrative permission. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **kinit** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

Files

/usr/lpp/ssp/bin/hmadm
Contains the **hmadm** command.

Related Information

File: **/spdata/sys1/spmon/hmacls**

hmcmds

Purpose

hmcmds – Controls the state of the SP hardware.

Syntax

hmcmds [–**a** | –**v**] [–**f** *file_name*] [–**u** *microcode_file_name*]
[–**G**] *command* [*slot_spec ...* | **all**]

Flags

–**a** Exits immediately after sending the **VFOP** command to the specified hardware; that is, it does not wait for the hardware state to match the command.

–**v** Specifies verbose mode. The percentage of hardware components whose state matches the **VFOP** command is displayed at five-second intervals. The following are also displayed:

- The number of components to which the **VFOP** command was sent
- A list of components which did not change state
- The number of components expected to change state
- The number of components which achieved the expected state
- The number of components to which the **VFOP** command was not sent
- A list of components to which the **VFOP** command was not sent

–**f** *file_name*
Uses *file_name* as the source of slot ID specifications.

–**u** *microcode_file_name*
Uses *microcode_file_name* as the source of supervisor microcode that is loaded to the specified *slot_spec*. If the *microcode_file_name* is not fully qualified, the file must be in the current directory. This option is allowed only with the **microcode** command.

–**G** Specifies Global mode. With this flag, commands can be sent to any hardware.

Operands

command Specifies the command to send to the hardware components.

slot_spec Specifies the addresses of the hardware components.

Description

Use this command to control the state of the SP hardware. Control is provided via the Virtual Front Operator Panel (VFOP). VFOP is a set of commands that can be sent to the hardware components contained in one or more SP frames. Each frame consists of 18 slots, numbered 0 through 17, where slot 0 represents the frame itself, slot 17 can contain a switch and slots 1 through 16 can contain thin or wide processing nodes. Wide nodes occupy two slots and are addressed by the

odd slot number. In a switch only frame, slots 1 through 16 can contain switches; the switches occupy two slots and are addressed by the even slot number.

Normally, commands are only sent to the hardware components in the current system partition. A system partition only contains processing nodes. The switches and the frames themselves are not contained in any system partition. To send **VFOP** commands to hardware components not in the current system partition or to any frame or switch, use the **–G** flag.

The following list describes the **VFOP** command set. Commands that require the **–G** flag are marked by an asterisk (*). Commands marked by a double asterisk (**) are primarily used by the **Eclock** command and are not intended for general use since an in-depth knowledge of switch clock topology is required to execute these commands in the proper sequence.

Before issuing these commands, refer to the "Using a Switch" chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide* for detailed descriptions.

High Performance Switch

extclk1 Sets the High Performance Switch clock multiplexor to the External Clock 1.**

extclk2 Sets the High Performance Switch clock multiplexor to the External Clock 2.**

extclk3 Sets the High Performance Switch clock multiplexor to the External Clock 3.**

intclk Sets the High Performance Switch clock multiplexer to the Internal Clock.**

SP Switch

clkdrv2 Sets the SP Switch clock drive to the Phase Lock Loop 2.**

clkdrv3 Sets the SP Switch clock drive to the Phase Lock Loop 3.**

clkdrv4 Sets the SP Switch clock drive to the Phase Lock Loop 4.**

clkdrv5 Sets the SP Switch clock drive to the Phase Lock Loop 5.**

hold_power_reset
Performs power-on reset of SP Switch and holds the SP Switch in reset state. Requires **rel_power_reset** to release.**

hold_synch_reset
Performs synchronous reset of SP Switch and holds the SP Switch in reset state. Requires **rel_synch_reset** to release.**

intclk2 Sets the SP Switch clock input to the Local Oscillator 2.**

intclk4 Sets the SP Switch clock input to the Local Oscillator 4.**

jack3 Sets the SP Switch clock input to the External Jack 3.**

jack4 Sets the SP Switch clock input to the External Jack 4.**

jack5 Sets the SP Switch clock input to the External Jack 5.**

jack6 Sets the SP Switch clock input to the External Jack 6.**

jack7 Sets the SP Switch clock input to the External Jack 7.**

jack8 Sets the SP Switch clock input to the External Jack 8.**

jack9 Sets the SP Switch clock input to the External Jack 9.**

jack10 Sets the SP Switch clock input to the External Jack 1.**

jack11 Sets the SP Switch clock input to the External Jack 11.**

jack12 Sets the SP Switch clock input to the External Jack 12.**

jack13 Sets the SP Switch clock input to the External Jack 13.**

jack14 Sets the SP Switch clock input to the External Jack 14.**

jack15 Sets the SP Switch clock input to the External Jack 15.**

jack16 Sets the SP Switch clock input to the External Jack 16.**

jack17 Sets the SP Switch clock input to the External Jack 17.**

jack18 Sets the SP Switch clock input to the External Jack 18.**

jack19 Sets the SP Switch clock input to the External Jack 19.**

jack20 Sets the SP Switch clock input to the External Jack 20.**

jack21 Sets the SP Switch clock input to the External Jack 21.**

jack22 Sets the SP Switch clock input to the External Jack 22.**

jack23 Sets the SP Switch clock input to the External Jack 23.**

jack24 Sets the SP Switch clock input to the External Jack 24.**

jack25 Sets the SP Switch clock input to the External Jack 25.**

jack26 Sets the SP Switch clock input to the External Jack 26.**

jack27 Sets the SP Switch clock input to the External Jack 27.**

jack28 Sets the SP Switch clock input to the External Jack 28.**

jack29 Sets the SP Switch clock input to the External Jack 29.**

jack30 Sets the SP Switch clock input to the External Jack 30.**

jack31 Sets the SP Switch clock input to the External Jack 31.**

jack32 Sets the SP Switch clock input to the External Jack 32.**

jack33 Sets the SP Switch clock input to the External Jack 33.**

jack34 Sets the SP Switch clock input to the External Jack 34.**

power_on_reset
Performs power-on reset of SP Switch. Includes chip self-test and synchronous reset.**

rel_power_reset
Releases SP Switch from **hold_power_reset** state.**

rel_synch_reset
Releases SP Switch from **hold_synch_reset** state.**

synch_reset
Performs synchronous reset of SP Switch. Turns off error enables and clears errors.**

Any Frame, Node, or Switch that Supports Microcode Download

basecode
Performs a power off of the node and switches the active frame, node, or switch supervisor to basecode mode causing the active supervisor to become nonactive and the basecode supervisor to become active.*

Note: You must issue this command before issuing the **microcode** command.

boot_application
Performs a boot of the frame, node, or switch basecode application and supervisor.*

exec_application
Causes the basecode to execute the nonactive frame, node, or switch supervisor thus making it active.*

microcode
Performs a download of supervisor microcode to the frame, node, or switch.*

Note: You must issue the **basecode** command before issuing this command.

rosdump Dumps the contents of the frame, node, or switch basecode or supervisor application, whichever is active. The contents are dumped to an aixterm that is opened for serial data read to the target slot.*

Refer to the **s1term** command for information on making serial connections.

Any Node

normal Sets the keylock on a processing node to the Normal position.

reset Presses and releases the reset button on a processing node.

secure Sets the keylock on a processing node to the Secure position.

service Sets the keylock on a processing node to the Service position.

Any Frame

runpost Initiates Power-On Self Tests (POST) in the frame supervisor.*

setid Sets the frame ID into the frame supervisor.*

Any Frame, Node, or Switch

off Disables power to the frame power supplies, a processing node, or a switch.

on Enables power to the frame power supplies, a processing node, or a switch.

Any Node or Switch

flash Flashes the I²C address of a processing node or a switch node in the node's yellow LED.

One of these commands must be specified using the *command* operand. The command is sent to the hardware specified by the *slot_spec* operands. However, the command is not sent to any hardware that is not in the current system partition unless the **–G** flag is specified. If the **–G** flag is not specified and the *slot_spec*

operands specify no hardware in the current system partition, an error message is displayed.

The *slot_spec* operands are interpreted as slot ID specifications. A slot ID specification names one or more slots in one or more SP frames and it has either of two forms:

```
fidlist:sidlist   or   nodlist
```

where:

fidlist = fval[,fval,...]

sidlist = sval[,sval,...]

nodlist = nval[,nval,...]

The first form specifies frame numbers and slot numbers. The second form specifies node numbers. A *fval* is a frame number or a range of frame numbers of the form *a–b*. A *sval* is a slot number from the set 0 through 17 or a range of slot numbers of the form *a–b*. A *nval* is a node number or a range of node numbers of the form *a–b*.

The relationship of node numbers to frame and slot numbers is shown in the following formula:

$node\_number = ((frame\_number - 1) \times 16) + slot\_number$

Note: Node numbers can only be used to specify slots 1 through 16 of any frame.

The following are some examples of slot ID specifications.

To specify slot 1 in frames 1 through 10, enter:

```
1-10:1
```

To specify frames 2, 4, 5, 6, and 7, enter:

```
2,4-7:0
```

To specify slots 9 through 16 in frame 5, enter:

```
5:9-16
```

If frame 5 contained wide nodes, the even slot numbers are ignored.

To specify specifies slots 1, 12, 13, 14, 15, and 16 in each of frames 3 and 4, enter:

```
3,4:1,12-16
```

To specify slot 17 in frame 4, enter:

```
4:17
```

To specify the nodes in slots 1 through 16 of frame 2, enter:

```
17-32
```

To specify the nodes in slot 1 of frame 1, slot 1 of frame 2 and slot 1 of frame 3, enter:

```
1,17,33
```

To specify the node in slot 6 of frame 1, enter:

```
6
```

Optionally, slot ID specifications can be provided in a file rather than as command operands. The file must contain one specification per line. The command requires that slot ID specifications be provided. If the command is to be sent to all SP hardware, the keyword **all** must be provided in lieu of the *slot_spec* operands. However, the **all** keyword can only be specified if the **–G** flag is specified and if the **VFOP** command is **on** or **off**, since **on** or **off** are the only commands common to all hardware components.

Commands sent to hardware for which they are not appropriate, or sent to hardware which does not exist, are silently ignored by the Hardware Monitor subsystem.

By default, and except for the **reset**, **flash**, and **run_post** commands, the **hmcmds** command does not terminate until the state of the hardware to which the command was sent matches the command or until 15 seconds have elapsed. If 15 seconds have elapsed, the **hmcmds** command terminates with a message stating the number of nodes whose state was expected to match the **VFOP** command sent and the number of nodes which actually are in that state. The state of hardware for which the **VFOP** command is inappropriate, or where the hardware does not exist, is ignored.

To execute the **hmcmds** command, the user must be authorized to access the Hardware Monitor subsystem and, for those frames specified to the command, the user must be granted **VFOP** permission. Commands sent to frames for which the user does not have **VFOP** permission are ignored. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **kinit** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

Files

/usr/lpp/ssp/bin/hmcmds
Contains the **hmcmds** command.

Related Information

Command: **hmmon**, **spsvrmgr**

Examples

1. To turn power off in all hardware, enter:

   ```
   hmcmds -G off all
   ```

2. In a five-frame SP system, to set the keyswitch on all processing nodes to Secure, enter:

   ```
   hmcmds secure 1-5:1-16
   ```

3. To set the clock multiplexor in the switches in frames 1 through 8 to external clock 3, enter:

   ```
   hmcmds -G extclk3 1-8:17
   ```

4. In a three-frame SP system, to set the keyswitch to Normal on node 6 and on the nodes in slot 2 of both frames 2 and 3, enter:

hmcmds

```
hmcmds normal 6 2,3:2
```

hmmon

Purpose

hmmon – Monitors the state of the SP hardware.

Syntax

hmmon [–**G**] [–**q**] [–**Q**] [–**r** | –**s**] [–**v** *var_nlist*]
[–**f** *file_name* | *slot_spec* ...]

hmmon –**V**

Flags

–**G** Specifies Global mode. With this flag, all hardware can be specified.

–**q** Displays the current state information prior to displaying changed state.

–**Q** Displays only the current state information and exits.

–**r** Displays the output in *raw* format.

–**s** Displays the output in *symbolic* format.

–**v** *var_nlist* Limits output to that of the state variables specified by *var_nlist*, a comma separated list of symbolic variable names. This list cannot contain blanks. Use the –**V** flag for a list of possible values.

–**V** Displays a descriptive list of symbolic variable names and variable indexes, and exits.

–**f** *file_name*
Uses the file *file_name* as the source of slot ID specifications.

Operands

slot_spec Displays the addresses of hardware components.

Description

Use this command to monitor the state of the SP hardware contained in one or more SP frames. Each frame consists of 18 slots, numbered 0 through 17, where slot 0 represents the frame itself, slot 17 can contain a switch and slots 1 through 16 can contain thin or wide processing nodes. Wide nodes occupy two slots and are addressed by the odd slot number. In a switch only frame, slots 1 through 16 can contain switches; the switches occupy two slots and are addressed by the even slot number.

With no flags and operands, the command prints to standard output descriptive text of all hardware state changes in the current system partition as they occur, from the time the command is invoked. The command does not terminate, unless the –**Q** flag or the –**V** flag is specified, and must be interrupted by the user. To monitor all of the hardware in the SP system, the –**G** flag must be specified. Note that the switches and the frames themselves are not contained in any system partition.

When one or more *slot_spec* operands are present, each operand is interpreted as a slot ID specification. A slot ID specification names one or more slots in one or more SP frames and it has either of two forms:

```
fidlist:[sidlist]   or   nodlist
```

where:

fidlist = fval[,fval,...]

sidlist = sval[,sval,...]

nodlist = nval[,nval,...]

The first form specifies frame numbers and slot numbers. The second form specifies node numbers. A *fval* is a frame number or a range of frame numbers of the form *a–b*. A *sval* is a slot number from the set 0 through 17 or a range of slot numbers of the form *a–b*. An *nval* is a node number or a range of node numbers of the form *a–b*. If a *sidlist* is not specified, all hardware in the frames specified by the *fidlist* is monitored.

The relationship of node numbers to frame and slot numbers is given by the following formula:

```
node_number = ((frame_number – 1) × 16) + slot_number
```

Note: The node numbers can only be used to specify slots 1 through 16 of any frame.

The following are some examples of slot ID specifications.

To specify all hardware in frames 1 through 10, enter:

```
1-10:
```

To specify frames 2, 4, 5, 6, and 7, enter:

```
2,4-7:0
```

To specify slots 9 through 16 in frame 5, enter:

```
5:9-16
```

If frame 5 contained wide nodes, the even slot numbers are ignored.

To specify slots 1, 12, 13, 14, 15, and 16 in each of frames 3 and 4, enter:

```
3,4:1,12-16
```

To specify slot 17 in frame 4, enter:

```
4:17
```

To specify the nodes in slots 1 through 16 of frame 2, enter:

```
17-32
```

To specify the nodes in slot 1 of frame 1, slot 1 of frame 2 and slot 1 of frame 3, enter:

```
1,17,33
```

To specify the node in slot 6 of frame 1, enter:

6

Optionally, slot ID specifications may be provided in a file rather than as command operands. The file must contain one specification per line. When slot ID specifications are provided to the command, only the hardware named by the specifications is monitored. Furthermore, of the hardware named by these specifications, only that which is located in the current system partition is monitored. To monitor hardware not contained in the current system partition, the **-G** flag must be specified. If the **-G** flag is not specified and the slot ID specifications name no hardware in the current system partition, an error message is displayed.

The default output displays hardware state information on a slot-by-slot basis. The state information for each slot is captioned by its frame ID and slot ID and consists of two columns. Each column contains state variable information, one variable per line. Each variable is displayed as descriptive text and a value. Boolean values are displayed as TRUE or FALSE. Integer values are displayed in hexadecimal.

The command provides two other output formats, raw and symbolic. Both write the information for one state variable per line. The *raw* format consists of four fields separated by white space as follows:

Field 1 Contains the frame ID.

Field 2 Contains the slot ID.

Field 3 Contains the variable ID in hexadecimal.

Field 4 Contains the variable value, as received from the hardware, in decimal.

The *symbolic* format consists of six fields separated by white space as follows:

Field 1 Contains the frame ID.

Field 2 Contains the slot ID.

Field 3 Contains the symbolic name of the state variable.

Field 4 Contains the variable value. Booleans are displayed as TRUE or FALSE. Integers are displayed as decimal values or floating point values, as appropriate to the definition of the variable.

Field 5 Contains the variable ID in hexadecimal.

Field 6 Contains the descriptive text for the variable. This is the same text that is displayed in the default output. Thus, “field” 6 contains embedded white space.

The alternative output formats are suitable for input to post-processing programs, such as **awk** or scripts.

Output in any format can be limited to display only information from the specified hardware that corresponds to a list of state variables supplied to the command with the **-v** flag.

To execute the **hmmon** command, the user must be authorized to access the Hardware Monitor subsystem and, for those frames specified to the command, the user must be granted “Monitor” permission. State information is not returned for frames for which the user does not have “Monitor” permission. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the

kinit command prior to executing this command. Alternatively, site specific procedures may be used to obtain the tokens that are otherwise obtained by **kinit**.

The user can monitor nonexistent nodes in an existing frame in order to detect when a node is added while the system is up and running. No information is returned for nonexistent nodes when the **–q** or **–Q** flag is specified.

Files

/usr/lpp/ssp/bin/hmmon
Contains the **hmmon** command.

Related Information

Command: **hmcmds**

Examples

The following is an example of default output:

```
frame 001, slot 00:
node 01 I2C not responding  FALSE  node 02 I2C not responding   TRUE
node 03 I2C not responding  FALSE  node 04 I2C not responding   TRUE
switch I2C not responding   FALSE  node 01 serial link open     TRUE
node 02 serial link open    FALSE  node 03 serial link open     TRUE
frame LED 1 (green)        0x0001  frame LED 2 (green)        0x0001
frame LED 3 (yellow)       0x0000  frame LED 4 (yellow)       0x0000
AC-DC section A power off   FALSE  AC-DC section B power off   FALSE
AC-DC section C power off   FALSE  AC-DC section D power off   FALSE
supervisor timer ticks     0x88f2  +48 voltage                0x0078
temperature                0x0036  supervisor serial number   0x1234
supervisor type            0x0011  supervisor code version    0x5ff5

frame 001, slot 01:
serial 1 DTR asserted        TRUE   -12 volt low warning      TRUE
-12 volt low shutdown       FALSE   -12 volt high warning     TRUE
+4 volt low shutdown        FALSE   +4 volt high warning      TRUE
fan 1 shutdown              FALSE   fan 2 warning             TRUE
DC-DC power on > 10 secs  TRUE   +5 DC-DC output good      TRUE
7 segment display flashing  FALSE   node/switch LED 1 (green) 0x0001
reset button depressed      FALSE   serial link open          TRUE
diagnosis return code      0x00dd   7 segment LED A           0x00ff
+5 I/O voltage             0x007f   +12 voltage               0x0096
```

The following is an example of *raw* output:

```
1 0 0x880f 32
1 0 0x881c 0
1 0 0x881d 4
1 0 0x8834 54
1 0 0x8839 4660
1 0 0x883a 17
1 0 0x88a8 1
1 1 0x9097 16
1 1 0x9098 0
1 1 0x9047 1
1 1 0x909d 128
1 1 0x9023 221
1 1 0x90a1 255
1 1 0x90a2 127
1 1 0x903b 24565
```

The following is an example of *symbolic* output:

```
1  0  nodefail1          FALSE    0x8802  node 01 I2C not responding
1  0  nodeLinkOpen1      TRUE     0x8813  node 01 serial link open
1  0  frACLED                 1  0x8824  frame LED 1 (green)
1  0  frNodeComm              0  0x8827  frame LED 4 (yellow)
1  0  frPowerOff_B       FALSE    0x882d  AC-DC section B power off
1  0  timeTicks           34881  0x8830  supervisor timer ticks
1  0  voltP48            46.800  0x8831  +48 voltage
1  0  type                   17  0x883a  supervisor type
1  0  codeVersion         24565  0x883b  supervisor code version
1  0  controllerResponds TRUE     0x88a8  Frame responding to polls
1  0  rs232DCD           TRUE     0x88a9  RS232 link DCD active
1  0  rs232CTS           TRUE     0x88aa  RS232 link CTS active
1  1  fanfail2           FALSE    0x9050  fan 2 shutdown
1  1  nodePowerOn10Sec   TRUE     0x904b  DC-DC power on > 10 secs
1  1  P5DCok             TRUE     0x9097  +5 DC-DC output good
1  1  powerLED                1  0x9047  node/switch LED 1 (green)
1  1  envLED                  0  0x9048  node/switch LED 2 (yellow)
1  1  keyModeSwitch           0  0x909b  key switch
1  1  serialLinkOpen     TRUE     0x909d  serial link open
1  1  LED7SegA              255  0x909f  7 segment LED A
1  1  voltP5i             4.978  0x90a2  +5 I/O voltage
```

The raw and symbolic formats output by the **hmmon** command contain the variable ID of each state variable. The following is the definition of the variable IDs and a definition of the value of each state variable.

Note: A Boolean is true when the Boolean value is nonzero and false when the Boolean value is zero.

```
/* variable IDs (VIDs) have the following format:
 *
 *  Bits   15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
 *
 *          1  t  t  t  t  t  v  v  v  v  v  v  v  v  v  v
 *
 *  where  ttttt = the variable prefix, as follows
 *         00000 = general variables - used for frame and node VIDs
 *         00001 = variables from SP1 thin nodes
 *         00010 = variables from the frame node
 *         00011 = variables from a switch node
 *         00100 = variables from a SP2 wide node
```

```
 *         00101 = variables from a SP2 thin node
 *
 *  and vvvvvvvvvv = the variable index.
 */

/* define variable base bit */

#define VID_BASE               0x8000

/* define variable prefix bits */

#define VID_PREFIX_GEN         0
#define VID_PREFIX_SP1_THIN    1
#define VID_PREFIX_FRAME       2
#define VID_PREFIX_SWITCH      3
#define VID_PREFIX_SP2_WIDE    4
#define VID_PREFIX_SP2_THIN    5
#define VID_PREFIX_MAX         VID_PREFIX_SP2_THIN

#define VIDP_GEN               (VID_PREFIX_GEN      << 10)
#define VIDP_SP1_THIN          (VID_PREFIX_SP1_THIN << 10)
#define VIDP_FRAME             (VID_PREFIX_FRAME    << 10)
#define VIDP_SWITCH            (VID_PREFIX_SWITCH   << 10)
#define VIDP_SP2_WIDE          (VID_PREFIX_SP2_WIDE << 10)
#define VIDP_SP2_THIN          (VID_PREFIX_SP2_THIN << 10)

/* define hardware supervisor types. These are the values returned
   in the card type state variables. They can be used to further
   identify the type of hardware in a given slot location */

#define SUP_SP1_THIN        0x21 /* supervisor in SP1 thin node */
#define SUP_FRAME           0x11 /* supervisor in frame node with
                                    PDU */
#define SUP_SEPBU_FRAME     0x12 /* supervisor in frame node with
                                    Scalable Electrical Power Based
                                    Unit */
#define SUP_FRAME_TT        0x14 /* supervisor in frame node with
                                    PDU and with twin-tail
                                    support */
#define SUP_SEPBU_FRAME_TT  0x16 /* supervisor in frame node with
                                    Scalable Electrical Power
                                    Based Unit and with twin-tail
                                    support */
#define SUP_SWITCH          0x31 /* supervisor in switch node */
#define SUP_SWITCH_PPP      0x32 /* supervisor in PPP switch node */
#define SUP_SP1_ETHIN       0x41 /* supervisor in early SP1 thin
                                    node */
#define SUP_SP2_WIDE_37V    0x53 /* supervisor in SP2 wide node
                                    with 3.7 volt power */
#define SUP_SP2_WIDE_40V    0x51 /* supervisor in SP2 wide node
                                    with 4.0 volt power */
#define SUP_SP2_THIN        0x61 /* supervisor in SP2 thin node */
#define SUP_SP2_EWIDE_37V   0x73 /* supervisor in early SP2 wide
                                    node with 3.7 volt power */
#define SUP_SP2_EWIDE_40V   0x71 /* supervisor in early SP2 wide
                                    node with 4.0 volt power */
#define SUP_SWITCH_TBS      0x81 /* tbs supervisor */
```

```
#define SUP_TBS_BASE        0x80 /* tbs supervisor - base packet */
#define SUP_PICO_BASE       0x10 /* pico frame - base packet */

/* A frame VID is defined by
 *
 *     VID_BASE + VIDP_GEN + 0x0000
 *
 * A node VID is defined by
 *
 *     VID_BASE + VIDP_GEN + 0x0001
 *
 * A state VID is defined by
 *
 *     VID_BASE + VIDP_SP1_THIN + index
 *     VID_BASE + VIDP_FRAME    + index
 *     VID_BASE + VIDP_SWITCH   + index
 *     VID_BASE + VIDP_SP2_WIDE + index
 *     VID_BASE + VIDP_SP2_THIN + index
 *
 * The VID_PREFIX_* prefix can be used to determine the type of
 * hardware which generated the state variable. The macro
 * VID_PREFIX(vid), with the VID as the argument, produces
 * VID_PREFIX_*. The macro VID_INDEX(vid), with the VID as the
 * argument, produces the variable index. Variables generated by
 * different hardware  types that have a common meaning have their
 * indices defined to the same variable index.
 */

#define VID_INDEX(x)    ((x) & 0x3ff)
#define VID_PREFIX(x)   (((x) & 0x7c00) >> 10)

/*****************************************************************/
/* VIDG_* are general variables                                  */
/*****************************************************************/

/* The following has a two byte integer value */

#define VIDG_FRAME_ID   ( VID_BASE + VIDP_GEN + 0x0000 )

/* The following has two, one byte integer values. The first byte
   value is the node id, the second byte value is the (unsigned)
   length of the following data. */

#define VIDG_NODE_ID    ( VID_BASE + VIDP_GEN + 0x0001 )

/*****************************************************************/
/* VIDF_* are the variable indices for data from the frame nodes */
/*****************************************************************/

/* The following have one byte Boolean values */

#define VIDF_I2C_COMM_STAT_1   0x0002  /* node  1 I2C not responding
                                          if TRUE */
#define VIDF_I2C_COMM_STAT_2   0x0003  /* node  2 I2C not responding
                                          if TRUE */
#define VIDF_I2C_COMM_STAT_3   0x0004  /* node  3 I2C not responding
                                          if TRUE */
#define VIDF_I2C_COMM_STAT_4   0x0005  /* node  4 I2C not responding
```

```
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_5   0x0006  /* node  5 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_6   0x0007  /* node  6 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_7   0x0008  /* node  7 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_8   0x0009  /* node  8 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_9   0x000a  /* node  9 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_10  0x000b  /* node 10 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_11  0x000c  /* node 11 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_12  0x000d  /* node 12 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_13  0x000e  /* node 13 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_14  0x000f  /* node 14 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_15  0x0010  /* node 15 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_16  0x0011  /* node 16 I2C not responding
                                        if TRUE */
#define VIDF_I2C_COMM_STAT_17  0x0012  /* switch  I2C not responding
                                        if TRUE */

#define VIDF_S1_STATUS_1       0x0013  /* node  1 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_2       0x0014  /* node  2 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_3       0x0015  /* node  3 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_4       0x0016  /* node  4 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_5       0x0017  /* node  5 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_6       0x0018  /* node  6 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_7       0x0019  /* node  7 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_8       0x001a  /* node  8 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_9       0x001b  /* node  9 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_10      0x001c  /* node 10 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_11      0x001d  /* node 11 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_12      0x001e  /* node 12 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_13      0x001f  /* node 13 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_14      0x0020  /* node 14 serial link open
                                        if TRUE */
#define VIDF_S1_STATUS_15      0x0021  /* node 15 serial link open
```

```
                                        if TRUE */
#define VIDF_S1_STATUS_16      0x0022  /* node 16 serial link open
                                        if TRUE */

/* The following have one byte integer values */

#define VIDF_DIAG              0x0023  /* diagnosis return code */

#define VIDF_LEDS_AC           0x0024  /* 1 = LED 1 green on, AC
                                          OK */
                                       /* 2 = LED 1 green blink, AC
                                          fail */
#define VIDF_LEDS_DC           0x0025  /* 1 = LED 2 green on, DC
                                          OK */
                                       /* 2 = LED 2 green blink, DC
                                          fail */
#define VIDF_LEDS_RS232        0x0026  /* 1 = LED 3 yellow on, RS232
                                          active */
                                       /* 2 = LED 3 yellow blink,
                                          RS232 error */
#define VIDF_LEDS_I2C_ERROR    0x0027  /* 0 = LED 4 yellow off, one
                                          or more nodes
                                          responding */
                                       /* 2 = LED 4 yellow blink,
                                          I2C error: 0 nodes
                                          responding (bus
                                          dead) */

/* The following have one byte Boolean values */

#define VIDF_ACDC_STAT_A       0x0028  /* DC section A failure if
                                          FALSE */
#define VIDF_ACDC_STAT_B       0x0029  /* DC section B failure if
                                          FALSE */
#define VIDF_ACDC_STAT_C       0x002a  /* DC section C failure if
                                          FALSE */
#define VIDF_ACDC_STAT_D       0x002b  /* DC section D failure if
                                          FALSE */

#define VIDF_ACDC_CTRL_A       0x002c  /* AC-DC section A power off
                                          if TRUE */
#define VIDF_ACDC_CTRL_B       0x002d  /* AC-DC section B power off
                                          if TRUE */
#define VIDF_ACDC_CTRL_C       0x002e  /* AC-DC section C power off
                                          if TRUE */
#define VIDF_ACDC_CTRL_D       0x002f  /* AC-DC section D power off
                                          if TRUE */

/* The following has a two byte integer value */

#define VIDF_TIMER             0x0030  /* number of ticks, 100ms
                                          tick */

/* The following have one byte integer values */

#define VIDF_VOLT_P48          0x0031  /* volts = value * 390.0mV */

                                       /* these are not currently
```

```
                                        implemented in hardware */
#if 0
#define VIDF_VOLT_P48BAT       0x0032  /* volts = value * 390.0mV */
#define VIDF_EXT_TEMP          0x0033  /* temp = value * 0.392 C */
#endif

#define VIDF_TEMP              0x0034  /* temp = value * 0.392 C */

#define VIDF_AMPA              0x0035  /* current = value * 0.391
                                          A */
#define VIDF_AMPB              0x0036  /* current = value * 0.391
                                          A */
#define VIDF_AMPC              0x0037  /* current = value * 0.391
                                          A */
#define VIDF_AMPD              0x0038  /* current = value * 0.391
                                          A */

/* The following has a two byte integer value */

#define VIDF_SN                0x0039  /* serial number */

/* The following has a one byte integer value */

#define VIDF_CARD_TYPE         0x003a  /* supervisor card type */

/* The following has a two byte integer value */

#define VIDF_VERSION           0x003b  /* code version number */

/* The following have one byte integer values */

#define VIDF_LEDS_POWER_MODS   0x003c  /* 1 = LED 1 green on, all
                                             modules present */
                                       /* 2 = LED 1 green blink, one
                                             or more modules not
                                             present */
#define VIDF_LEDS_DC_POWER     0x003d  /* 1 = LED 2 green on, DC
                                             power good */
                                       /* 2 = LED 2 green blink, one
                                             or more modules DC
                                             power fail */

/* The following have one byte Boolean values */

#define VIDF_ACDC_GOOD_A       0x003e  /* Module A power good if
                                          FALSE */
#define VIDF_ACDC_GOOD_B       0x003f  /* Module B power good if
                                          FALSE */
#define VIDF_ACDC_GOOD_C       0x0040  /* Module C power good if
                                          FALSE */
#define VIDF_ACDC_GOOD_D       0x0041  /* Module D power good if
                                          FALSE */
#define VIDF_ACDC_PRESENT_A    0x0042  /* Module A present if
                                          FALSE */
#define VIDF_ACDC_PRESENT_B    0x0043  /* Module B present if
                                          FALSE */
#define VIDF_ACDC_PRESENT_C    0x0044  /* Module C present if
```

```
                                        FALSE */
#define VIDF_ACDC_PRESENT_D    0x0045  /* Module D present if
                                        FALSE */

#define VIDF_ACDC_CTRL         0x0046  /* Frame power off if TRUE */

/* The following has a two byte integer value */

#define VIDF_POLL_RATE         0x0067  /* Daemon poll rate in
                                           seconds */

/* The following have one byte Boolean values */

#define VIDF_VOLT_P48_RANGE    0x0083  /* +48 value out of range */
#define VIDF_TEMP_RANGE        0x0084  /* temp out of range */
#define VIDF_AMPA_RANGE        0x0085  /* current out of range */
#define VIDF_AMPB_RANGE        0x0086  /* current out of range */
#define VIDF_AMPC_RANGE        0x008b  /* current out of range */
#define VIDF_AMPD_RANGE        0x008c  /* current out of range */
#define VIDF_FRAME_RESPONDING  0x00a8  /* Frame responding to polls
                                        if TRUE */
#define VIDF_RS232_DCD         0x00a9  /* RS232 link DCD active if
                                        TRUE */
#define VIDF_RS232_CTS         0x00aa  /* RS232 link CTS active if
                                        TRUE */
#define VIDF_FIRST_FRAME_PKT   0x00ab  /* first packet received
                                        from frame only returned
                                        as state change, not
                                        returned on a query */

| /* The following variable indices are for frame supervisor types
|    0x14, 0x16 */

/* The following have one byte Boolean values */

#define VIDF_FRAME_ID_MISMATCH 0x0071  /* frame ID in hardware does
                                        not match configured frame
                                        ID if TRUE */
#define VIDF_FRAME_TAIL_ACTIVE 0x0077  /* monitor connection to
                                        frame supervisor is
                                        "active", i.e. the frame
                                        supervisor will take
                                        hardware control commands
                                        from the monitor, if
                                        TRUE */

/* The following have one byte integer values */

#define VIDF_FRAME_TAIL        0x0078  /* 1 = monitor connected via
                                           S1 to frame
                                           supervisor */
                                       /* 2 = monitor connected via
                                           S2 to frame
                                           supervisor */
#define VIDF_LEDS_TAIL1        0x0079  /* 1 = LED 4 green on, S1
                                           tail active */
#define VIDF_LEDS_TAIL2        0x007a  /* 1 = LED 3 green on, S2
```

```
                                                    tail active */
#define VIDF_LEDS_RS232_S1     0x0096  /* 1 = LED 8 yellow on, RS232
                                          active */
                                       /* 2 = LED 8 yellow blink,
                                          RS232 error */
#define VIDF_LEDS_RS232_S2     0x0099  /* 1 = LED 7 yellow on, RS232
                                          active */
                                       /* 2 = LED 7 yellow blink,
                                          RS232 error */
#define VIDF_LEDS_I2C_ERROR_TT 0x009a  /* 0 = LED 6 yellow off, one
                                          or more nodes
                                          responding */
                                       /* 2 = LED 6 yellow blink,
                                          I2C error: 0 nodes
                                          responding (bus
                                          dead) */

| /* The following variable indices are for frame type 0x10, which
|    is the basecode type for frame supervisor types 0x14, 0x16 */

| /* The following have one byte Boolean values */

| #define VIDF_FEEPROM_EMPTY     0x00bc  /* TRUE = empty flash after
|                                           download */
| #define VIDF_CHECKSUM_ERROR    0x00bd  /* TRUE = error during
|                                           download */
| #define VIDF_KEYID_MISMATCH    0x00be  /* TRUE = wrong supervisor
|                                           type download
|                                           attempt */
| #define VIDF_SUP_FAILURE       0x00bf  /* TRUE = supervisor
|                                           execution failed
|                                           after download */
| #define VIDF_SUBTYPE           0x00d3  /* hardware subtype */

| /* The following have one byte integer values */

| #define VIDF_FEEPROM_ERASE_COUNT   0x00da  /* erase count
|                                               accumulator */
| #define VIDF_SUP_DATA_CHECKSUM     0x00dd  /* supervisor
|                                               checksums */
| #define VIDF_SUP_ROM_CHECKSUM      0x00de  /* supervisor
|                                               checksums */
| #define VIDF_SOFTWARE_KEYID        0x00db  /* supervisor keyid */
| #define VIDF_HARDWARE_KEYID        0x00dc  /* hardware keyid */
| #define VIDF_BOARD_ID              0x00e5  /* board id */
| #define VIDF_SUP_ID                0x00e6  /* supervisor id */

| /* The following has a 2 byte integer value */

| #define VIDF_APPLICATION_LEVEL     0x00df  /* software level of
|                                               loaded app */

/*****************************************************************/
/* VIDS_* are the variable indices for data from the switch nodes */
/*****************************************************************/

/* The following have one byte integer values */
```

```
#define VIDS_LEDS_POWER        0x0047  /* 1 = LED 1 green on, power
                                          OK */
                                       /* 2 = LED 1 green blink,
                                          DC-DC off */
                                       /* 0 = LED 1 green off, P48
                                          off */
#define VIDS_LEDS_ENVIRON      0x0048  /* 1 = LED 2 yellow on,
                                          warning */
                                       /* 2 = LED 2 yellow blink,
                                          critical error
                                          detected, power
                                          off */
                                       /* 0 = LED 2 yellow off,
                                          OK */

/* The following have one byte Boolean values */

#define VIDS_POWER_P48V        0x0049  /* +48V on if TRUE */
#define VIDS_POWER_DCDC        0x004a  /* DC-DC on if TRUE */
#define VIDS_POWER_DCDC_G10    0x004b  /* DC-DC on > 10 seconds
                                          if TRUE */
#define VIDS_POWER_OFF         0x004c  /* power off > 10 seconds
                                          if TRUE */

#define VIDS_FAN_WARNING_1     0x004d  /* Fan 1 warning if TRUE */
#define VIDS_FAN_SHUTDOWN_1    0x004e  /* Fan 1 shutdown if TRUE */
#define VIDS_FAN_WARNING_2     0x004f  /* Fan 2 warning if TRUE */
#define VIDS_FAN_SHUTDOWN_2    0x0050  /* Fan 2 shutdown if TRUE */
#define VIDS_FAN_WARNING_3     0x0051  /* Fan 3 warning if TRUE */
#define VIDS_FAN_SHUTDOWN_3    0x0052  /* Fan 3 shutdown if TRUE */
#define VIDS_FAN_WARNING_4     0x0053  /* Fan 4 warning if TRUE */
#define VIDS_FAN_SHUTDOWN_4    0x0054  /* Fan 4 shutdown if TRUE */

#define VIDS_FAN_WARNING_5     0x0055  /* Fan 5 warning if TRUE */
#define VIDS_FAN_SHUTDOWN_5    0x0056  /* Fan 5 shutdown if TRUE */

/* The following has a one byte integer value */

#define VIDS_MUX               0x0057  /* values are SW_INTCLK,
                                          SW_EXTCLK1, SW_EXTCLK2,
                                          SW_EXTCLK3 */

/* The following have one byte Boolean values */

#define VIDS_TEMP_WARN             0x0058  /* temp warning if
                                              TRUE */
#define VIDS_TEMP_SHUTDOWN         0x0059  /* temp shutdown if
                                              TRUE */

#define VIDS_POWER_WARN_P5H        0x005a  /* warning +5 high if
                                              TRUE */
#define VIDS_POWER_SHUTDOWN_P5H    0x005b  /* shutdown +5 high if
                                              TRUE */
#define VIDS_POWER_WARN_P5L        0x005c  /* warning +5 low if
                                              TRUE */
#define VIDS_POWER_SHUTDOWN_P5L    0x005d  /* shutdown +5 low if
                                              TRUE */
#define VIDS_POWER_WARN_N5H        0x005e  /* warning -5 high if
```

```
                                                TRUE */
#define VIDS_POWER_SHUTDOWN_N5H    0x005f  /* shutdown -5 high if
                                                TRUE */
#define VIDS_POWER_WARN_N5L        0x0060  /* warning -5 low if
                                                TRUE */
#define VIDS_POWER_SHUTDOWN_N5L    0x0061  /* shutdown -5 low if
                                                TRUE */


#define VIDS_POWER_WARN_P12H       0x0062  /* warning +12 high if
                                                TRUE */
#define VIDS_POWER_SHUTDOWN_P12H   0x0063  /* shutdown +12 high if
                                                TRUE */
#define VIDS_POWER_WARN_P12L       0x0064  /* warning +12 low if
                                                TRUE */
#define VIDS_POWER_SHUTDOWN_P12L   0x0065  /* shutdown +12 low if
                                                TRUE */
#define VIDS_POWER_SHUTDOWN_P48L   0x0066  /* shutdown +48 low if
                                                TRUE */

/* The following have one byte integer values */

#define VIDS_DIAG              VIDF_DIAG

#define VIDS_VOLTP5            0x0068  /* volt = value * 39.2mV */
#define VIDS_VOLTP12           0x0069  /* volt = value * 78.4mV */
#define VIDS_VOLTN5            0x006a  /* volt = value * -39.2mV */
#define VIDS_VOLTP48           VIDF_VOLT_P48

#define VIDS_TEMP              VIDF_TEMP

#if 0
#define VIDS_AMPP48            0x006d  /* amps = value * 19.6ma */
#endif

/* The following has a two byte integer value */

#define VIDS_TIMER             VIDF_TIMER

/* The following has a two byte integer value */

#define VIDS_SN                VIDF_SN

/* The following has a one byte integer value */

#define VIDS_CARD_TYPE         VIDF_CARD_TYPE

/* The following has a two byte integer value */

#define VIDS_VERSION           VIDF_VERSION

/* The following have one byte Boolean values */

#define VIDS_VOLTP5_RANGE      0x008d  /* volt out of range */
#define VIDS_VOLTP12_RANGE     0x008e  /* volt out of range */
#define VIDS_VOLTN5_RANGE      0x008f  /* volt out of range */
#define VIDS_VOLTP48_RANGE     VIDF_VOLT_P48_RANGE
#define VIDS_TEMP_RANGE        VIDF_TEMP_RANGE
```

```
/* The following variable indices are for frame type 0x80, which is
   the basecode type for frame supervisor type 0x81 */

/* The following have one byte Boolean values */

#define VIDS_FEEPROM_EMPTY          VIDF_FEEPROM_EMPTY
#define VIDS_CHECKSUM_ERROR         VIDF_CHECKSUM_ERROR
#define VIDS_KEYID_MISMATCH         VIDF_KEYID_MISMATCH
#define VIDS_SUP_FAILURE            VIDF_SUP_FAILURE
#define VIDS_SUBTYPE                VIDF_SUBTYPE

/* The following have one byte integer values */

#define VIDS_FEEPROM_ERASE_COUNT    VIDF_FEEPROM_ERASE_COUNT
#define VIDS_SUP_DATA_CHECKSUM      VIDF_SUP_DATA_CHECKSUM
#define VIDS_SUP_ROM_CHECKSUM       VIDF_SUP_ROM_CHECKSUM
#define VIDS_SOFTWARE_KEYID         VIDF_SOFTWARE_KEYID
#define VIDS_HARDWARE_KEYID         VIDF_HARDWARE_KEYID
#define VIDS_BOARD_ID               VIDF_BOARD_ID
#define VIDS_SUP_ID                 VIDF_SUP_ID

/* The following has a 2 byte integer value */

#define VIDS_APPLICATION_LEVEL      VIDF_APPLICATION_LEVEL

/* The following variable indices are for frame supervisor type
   0x81 */

/* The following have one byte Boolean values */

#define VIDS_CHIP_SELFTEST_ACTIVE   0x00c0  /* chip selftest active
                                               if TRUE */
#define VIDS_CHIP_RECEIVED_INIT     0x00c1  /* chip received init if
                                               TRUE */
#define VIDS_CHIP_CLOCK_ACTIVE      0x00c2  /* chip clock active if
                                               TRUE */
#define VIDS_PORT_REC_CLOCK_ACTIVE  0x00c3  /* receive port active
                                               if TRUE */
#define VIDS_PORT_SEND_TUNED        0x00c4  /* send port tuned if
                                               TRUE */
#define VIDS_PORT_REC_TUNED         0x00c5  /* receive port tuned if
                                               TRUE */
#define VIDS_SYNCH_RESET            0x00c6  /* synch reset active if
                                               TRUE */
#define VIDS_POWER_ON_RESET         0x00c7  /* power on reset active
                                               if TRUE */
#define VIDS_PS1_FAILURE            0x00c8  /* power supply 1
                                               failure if TRUE */
#define VIDS_PS2_FAILURE            0x00c9  /* power supply 2
                                               failure if TRUE */
#define VIDS_OVERCURRENT_WARNING    0x00ca  /* overcurrent warning
                                               if TRUE */
#define VIDS_OVERCURRENT_SHUTDOWN   0x00cb  /* overcurrent shutdown
                                               if TRUE */
#define VIDS_PS_PARALLEL_FAILURE    0x00cc  /* parallel failure if
                                               TRUE */
#define VIDS_EPROM_PROGRAM          0x00cd  /* program EPROM
```

```
                                                        complete if TRUE */
#define VIDS_EPROM_ERASE           0x00ce  /* erase EPROM complete
                                              if TRUE */
#define VIDS_POWER_WARN_P3_3H      0x00cf  /* warning +3.3 high if
                                              TRUE */
#define VIDS_POWER_SHUTDOWN_P3_3H  0x00d0  /* shutdown +3.3 high if
                                              TRUE */
#define VIDS_POWER_WARN_P3_3L      0x00d1  /* warning +3.3 low if
                                              TRUE */
#define VIDS_POWER_SHUTDOWN_P3_3L  0x00d2  /* shutdown +3.3 low if
                                              TRUE */
#define VIDS_TBS_SUBTYPE           VIDS_SUBTYPE /* TRUE=TBS;
                                                   FALSE=TBS-LC8 */
#define VIDS_PS_PARALLEL_RANGE     0x00d4  /* parallel out of
                                              range */
#define VIDS_PS1_POWER_GOOD_RANGE  0x00d5  /* ps1 power out of
                                              range */
#define VIDS_PS2_POWER_GOOD_RANGE  0x00d6  /* ps2 power out of
                                              range */
#define VIDS_PS1_FUSE_GOOD_RANGE   0x00d7  /* ps1 fuse out of
                                              range */
#define VIDS_PS2_FUSE_GOOD_RANGE   0x00d8  /* ps2 fuse out of
                                              range */
#define VIDS_VOLT_P3_3_RANGE       0x00d9  /* +3.3 voltage out of
                                              range */

/* The following have one byte integer values */

#define VIDS_CLOCK_SOURCE          0x00e0  /* current clock
                                              source */
#define VIDS_OSC                   0x00e1  /* current oscillator */
#define VIDS_PLL                   0x00e2  /* current phase lock
                                              loop */
#define VIDS_CHIP                  0x00e3  /* current chip */
#define VIDS_PORT                  0x00e4  /* current port */
#define VIDS_TBS_BOARD_ID          VIDS_BOARD_ID
#define VIDS_TBS_SUP_ID            VIDS_SUP_ID
#define VIDS_POWER_SUPPLY_1        0x00e7  /* power supply #1 */
#define VIDS_POWER_SUPPLY_2        0x00e8  /* power supply #2 */
#define VIDS_PS_PARALLEL           0x00e9  /* power supply parallel
                                              voltage */
#define VIDS_PS1_POWER_GOOD        0x00ea  /* power supply #1;
                                              power voltage */
#define VIDS_PS2_POWER_GOOD        0x00eb  /* power supply #2;
                                              power voltage */
#define VIDS_PS1_FUSE_GOOD         0x00ec  /* power supply #1; fuse
                                              voltage */
#define VIDS_PS2_FUSE_GOOD         0x00ed  /* power supply #2; fuse
                                              voltage */
#define VIDS_VOLT_P3_3             0x00ee  /* +3.3 card voltage */

/*****************************************************************/
/* VIDN_* are the variable indices for data from the processing  */
/* nodes                                                         */
/*****************************************************************/

/* The following have one byte Boolean values */
```

```
#define VIDN_S1_DTR                0x0072  /* serial 1 DTR if
                                              TRUE */

#define VIDN_POWER_WARN_N12L       0x0073  /* warning -12 low if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_N12L   0x0074  /* shutdown -12 low if
                                              TRUE */
#define VIDN_POWER_WARN_N12H       0x0075  /* warning -12 high if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_N12H   0x0076  /* shutdown -12 high if
                                              TRUE */
#define VIDN_POWER_WARN_P12L       VIDS_POWER_WARN_P12L
#define VIDN_POWER_SHUTDOWN_P12L   VIDS_POWER_SHUTDOWN_P12L
#define VIDN_POWER_WARN_P12H       VIDS_POWER_WARN_P12H
#define VIDN_POWER_SHUTDOWN_P12H   VIDS_POWER_SHUTDOWN_P12H

#define VIDN_POWER_WARN_P5ML       0x007b  /* warning +5m low if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P5ML   0x007c  /* shutdown +5m low if
                                              TRUE */
#define VIDN_POWER_WARN_P5MH       0x007d  /* warning +5m high if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P5MH   0x007e  /* shutdown +5m high if
                                              TRUE */
#define VIDN_POWER_WARN_P5IL       0x007f  /* warning +5i low if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P5IL   0x0080  /* shutdown +5i low if
                                              TRUE */
#define VIDN_POWER_WARN_P5IH       0x0081  /* warning +5i high if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P5IH   0x0082  /* shutdown +5i high if
                                              TRUE */

#define VIDN_FAN_WARNING_5         VIDS_FAN_WARNING_5
#define VIDN_FAN_SHUTDOWN_5        VIDS_FAN_SHUTDOWN_5
#define VIDN_TEMP_WARN             VIDS_TEMP_WARN
#define VIDN_TEMP_SHUTDOWN         VIDS_TEMP_SHUTDOWN
#define VIDN_POWER_WARN_P4L        0x0087  /* warning +4 low if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P4L    0x0088  /* shutdown +4 low if
                                              TRUE */
#define VIDN_POWER_WARN_P4H        0x0089  /* warning +4 high if
                                              TRUE */
#define VIDN_POWER_SHUTDOWN_P4H    0x008a  /* shutdown +4 high if
                                              TRUE */

#define VIDN_FAN_WARNING_1         VIDS_FAN_WARNING_1
#define VIDN_FAN_SHUTDOWN_1        VIDS_FAN_SHUTDOWN_1
#define VIDN_FAN_WARNING_2         VIDS_FAN_WARNING_2
#define VIDN_FAN_SHUTDOWN_2        VIDS_FAN_SHUTDOWN_2
#define VIDN_FAN_WARNING_3         VIDS_FAN_WARNING_3
#define VIDN_FAN_SHUTDOWN_3        VIDS_FAN_SHUTDOWN_3
#define VIDN_FAN_WARNING_4         VIDS_FAN_WARNING_4
#define VIDN_FAN_SHUTDOWN_4        VIDS_FAN_SHUTDOWN_4

#define VIDN_POWER_P48V            VIDS_POWER_P48V
#define VIDN_POWER_OFF             VIDS_POWER_OFF
```

```
#define VIDN_POWER_DCDC             VIDS_POWER_DCDC
#define VIDN_POWER_DCDC_G10         VIDS_POWER_DCDC_G10

#define VIDN_P5_ON              0x0097  /* P5 DC-DC output good if
                                           TRUE */
#define VIDN_7SEG_FLASHING      0x0098  /* 7 segment display
                                           flashing if TRUE */

/* The following have one byte integer values */

#define VIDN_LEDS_POWER         VIDS_LEDS_POWER
#define VIDN_LEDS_ENVIRON       VIDS_LEDS_ENVIRON
#define VIDN_KEY_SWITCH         0x009b  /* 1 = secure mode */
                                        /* 2 = service mode */
                                        /* 0 = normal mode */

/* The following have one byte Boolean values */

#define VIDN_RESET_ACTIVE       0x009c  /* reset button depressed
                                           if TRUE */
#define VIDN_S1_LINK_OPEN       0x009d  /* serial link open if
                                           TRUE */

/* The following have one byte integer values */

#define VIDN_DIAG               VIDF_DIAG

#define VIDN_7SEG_LED_A         0x009f  /* raw segments for 7
                                           segment LED A */
#define VIDN_7SEG_LED_B         0x00a0  /* raw segments for 7
                                           segment LED B */
#define VIDN_7SEG_LED_C         0x00a1  /* raw segments for 7
                                           segment LED C */

#define VIDN_VOLTP5I            0x00a2  /* volt = value * 39.2mV */
#define VIDN_VOLTP12            VIDS_VOLTP12
#define VIDN_VOLTN12            0x00a4  /* volt = value *
                                                  -78.4mV */
                                        /* volt = value *
                                                  78.4mV -15V
                                           if card type is
                                           SUP_SP1_ETHIN */
#define VIDN_VOLTP5M            0x00a5  /* volt = value * 39.2mV */

#define VIDN_TEMP               VIDF_TEMP

#define VIDN_VOLTP4             0x00a7  /* volt = value *
                                                  39.2mV */

/* The following has a two byte integer value */

#define VIDN_TIMER              VIDF_TIMER

/* The following has a two byte integer value */

#define VIDN_SN                 VIDF_SN

/* The following has a one byte integer value */
```

```
#define VIDN_CARD_TYPE          VIDF_CARD_TYPE

/* The following has a two byte integer value */

#define VIDN_VERSION            VIDF_VERSION

/* the following has a variable length byte string value. The
   first byte of the string is an unsigned integer specifying the
   length of the remaining bytes in the string */

#define VIDN_S1DATA             0x00ac  /* S1 serial port data */
#define VIDF_S1DATA             VIDN_S1DATA
#define VIDS_S1DATA             VIDN_S1DATA

/* The following have one byte Boolean values */

#define VIDN_MEMORY_PROTECT         0x00ad  /* memory protect if
                                               TRUE, i.e.
                                               incompatible memory
                                               card installed in the
                                               node */
#define VIDN_CURRENT_SHUTDOWN       0x00ae  /* current shutdown if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P48L    VIDS_POWER_SHUTDOWN_P48L

#define VIDN_POWER_WARN_P5L         VIDS_POWER_WARN_P5L
#define VIDN_POWER_SHUTDOWN_P5L     VIDS_POWER_SHUTDOWN_P5L
#define VIDN_POWER_WARN_P5H         VIDS_POWER_WARN_P5H
#define VIDN_POWER_SHUTDOWN_P5H     VIDS_POWER_SHUTDOWN_P5H

/* The following have one byte integer value */

#define VIDN_VOLTP5                 VIDS_VOLTP5
#define VIDN_VOLTP48                VIDS_VOLTP48

/* The following have one byte Boolean values */

#define VIDN_POWER_WARN_P4DL        0x00b0  /* warning +4d low if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P4DL    0x00b1  /* shutdown +4d low if
                                               TRUE */
#define VIDN_POWER_WARN_P4DH        0x00b2  /* warning +4d high if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P4DH    0x00b3  /* shutdown +4d high if
                                               TRUE */
#define VIDN_POWER_WARN_P5DL        0x00b4  /* warning +5d low if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P5DL    0x00b5  /* shutdown +5d low if
                                               TRUE */
#define VIDN_POWER_WARN_P5DH        0x00b6  /* warning +5d high if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P5DH    0x00b7  /* shutdown +5d high if
                                               TRUE */
#define VIDN_POWER_WARN_P2_5DL      0x00f8  /* warning +2.5d low if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P2_5DL  0x00f9  /* shutdown +2.5d low if
```

```
                                                    TRUE */
#define VIDN_POWER_WARN_P2_5DH      0x00fa  /* warning +2.5d high if
                                               TRUE */
#define VIDN_POWER_SHUTDOWN_P2_5DH  0x00fb  /* shutdown +2.5d high
                                               if TRUE */

#define VIDN_POWER_5D_PRESENT       0x00b8  /* daughter card 5 volt
                                               supply installed
                                               if TRUE */
#define VIDN_POWER_4D_PRESENT       0x00b9  /* daughter card 4 volt
                                               supply installed
                                               if TRUE */
#define VIDN_POWER_2_5D_PRESENT     0x00fc  /* daughter card 2.5
                                               volt supply
                                               installed if TRUE */

/* The following have one byte integer value */

#define VIDN_VOLTP4D            0x00ba  /* volt = value * 29.3mV */
#define VIDN_VOLTP5D            0x00bb  /* volt = value * 39.2mV */
#define VIDN_VOLTP2_5D          0x00fd  /* volt = value * 39.2mV */

/* The following have one byte Boolean values */

#define VIDN_FAN_DSHUTDOWN_1    0x006b  /* Fan 1 delayed shutdown if
                                           TRUE */
#define VIDN_FAN_DSHUTDOWN_2    0x006c  /* Fan 2 delayed shutdown if
                                           TRUE */
#define VIDN_FAN_DSHUTDOWN_3    0x006e  /* Fan 3 delayed shutdown if
                                           TRUE */
#define VIDN_FAN_DSHUTDOWN_4    0x006f  /* Fan 4 delayed shutdown if
                                           TRUE */
#define VIDN_FAN_DSHUTDOWN_5    0x0070  /* Fan 5 delayed shutdown if
                                           TRUE */

#define VIDN_VOLTP5I_RANGE      0x0090  /* volt out of range */
#define VIDN_VOLTP12_RANGE      VIDS_VOLTP12_RANGE
#define VIDN_VOLTN12_RANGE      0x0091  /* volt out of range */
#define VIDN_VOLTP5M_RANGE      0x0092  /* volt out of range */
#define VIDN_TEMP_RANGE         VIDF_TEMP_RANGE
#define VIDN_VOLTP4_RANGE       0x0093  /* volt out of range */
#define VIDN_VOLTP5_RANGE       VIDS_VOLTP5_RANGE
#define VIDN_VOLTP48_RANGE      VIDS_VOLTP48_RANGE
#define VIDN_VOLTP4D_RANGE      0x0094  /* volt out of range */
#define VIDN_VOLTP5D_RANGE      0x0095  /* volt out of range */
#define VIDN_VOLTP2_5D_RANGE    0x00fe  /* volt out of range */

/* The following variable indices are for node type 0x10, which is
   the basecode type for node supervisor types 0x14, 0x16 */

/* The following have one byte Boolean values */

#define VIDN_FEEPROM_EMPTY          VIDF_FEEPROM_EMPTY
#define VIDN_CHECKSUM_ERROR         VIDF_CHECKSUM_ERROR
#define VIDN_KEYID_MISMATCH         VIDF_KEYID_MISMATCH
#define VIDN_SUP_FAILURE            VIDF_SUP_FAILURE
#define VIDN_SUBTYPE                VIDF_SUBTYPE
```

```
                    /* The following have one byte integer values */

                    #define VIDN_FEEPROM_ERASE_COUNT    VIDF_FEEPROM_ERASE_COUNT
                    #define VIDN_SUP_DATA_CHECKSUM      VIDF_SUP_DATA_CHECKSUM
                    #define VIDN_SUP_ROM_CHECKSUM       VIDF_SUP_ROM_CHECKSUM
                    #define VIDN_SOFTWARE_KEYID         VIDF_SOFTWARE_KEYID
                    #define VIDN_HARDWARE_KEYID         VIDF_HARDWARE_KEYID
                    #define VIDN_BOARD_ID               VIDF_BOARD_ID
                    #define VIDN_SUP_ID                 VIDF_SUP_ID

                    /* The following has a 2 byte integer value */

                    #define VIDN_APPLICATION_LEVEL      VIDF_APPLICATION_LEVEL

                    /* The following variable indices are for node type 0xa1, which is
                       the sepbu supervisor used in an smp node. */

                    /* The following have one byte Boolean values */

                    #define VIDN_POWER_SWITCH_STATUS  0x00f0  /* power switch status */
                    #define VIDN_LCD_1_FLASH          0x00f1  /* LCD line 1 is
                                                                 flashing */
                    #define VIDN_LCD_2_FLASH          0x00f2  /* LCD line 2 is
                                                                 flashing */
                    #define VIDN_SMP_POWER_LED        0x00f6  /* SMP LED 1 (green) */
                    #define VIDN_SMP_DIAG_LED         0x00f7  /* SMP LED 2 (yellow) */

                    /* The following have one byte integer values */

                    #define VIDN_HARDWARE_STATUS      0x00f3  /* hardware status byte */

                    /* The following has a variable length byte string value. The first
                       byte of the string is an unsigned integer specifying the length
                       of the remaining bytes in the string. The string is NULL ('\0')
                       terminated. */

                    #define VIDN_LCD_1                0x00f4  /* LCD line 1
                                                                 (chars 1-16) */
                    #define VIDN_LCD_2                0x00f5  /* LCD line 2
                                                                 (chars 17-32) */
```

hmreinit

Purpose

hmreinit – Stops and starts the Hardware Monitor daemon and modifies the System Data Repository (SDR) as necessary.

Syntax

hmreinit

Flags

None.

Operands

None.

Description

Use this command to reinitialize the Hardware Monitor daemon when changes to the SP system occur. Specifically, **hmreinit** determines if there are any changes in the switch configuration (such as, adding, deleting, or upgrading switches). The **hmreinit** command then calls **SDR_config** –**u** to update the switch information in the SDR and generates switch node numbers based on this change. If a switch configuration change is detected, **hmreinit** will test to see if a single system partition exists. If only one system partition exists, **hmreinit** will delete the Syspar_map entries from the SDR and then calls **create_par_map** to generate the correct objects. If more than one system partition exists, **hmreinit** will issue a message to that effect and exits.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that more than one system partition was found.

Security

You must have root privilege to run this command and have a valid ticket.

Implementation Specifics

This command is part of the Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/install/bin/hmreinit

Related Information

Commands: **spframe**, **SDR_config**

For additional information, refer to the “Reconfiguring the IBM RS/6000 SP System” chapter in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*.

hostlist

Purpose

hostlist – Lists SP host names to standard output based on criteria.

Syntax

hostlist [–**s** *framerange*:*slotrange*] [–**f** *file_name*] [–**a**] [–**G**]
[–**n** *noderange*] [–**w** *host_name*,*host_name*, ...]
[–**e** *host_name*,*host_name*, ...] [–**v**] [–**d** | –**l**]
[–**N** *node_group*,*node_group*, ...]

Flags

–**s** Specifies a range of frames and a range of slots on each of the frames. Ranges are specified as in 1–3, meaning 1 through 3 inclusive, and as 1,3,15, meaning 1, 3, and 15. Ranges can incorporate both styles as in 1–10,15. So, 1–3,5:1–2,4 would refer to slots 1,2 and 4 on each of the frames 1,2,3, and 5. If a node occupies more than one slot, referring to either or both of the slots refers to the node.

–**f** Specifies the file name of a working collective file as in the **dsh** working collective, containing a host name on each line. This can be in the format of a Parallel Operating Environment (POE) **host.list** file.

–**a** Specifies that the System Data Repository (SDR) **initial_hostname** field for all nodes in the current system partition be written to standard output. For each node, this corresponds to what the **hostname** command returns on the node.

–**G** Changes the scope of the arguments associated with the –**a**, –**n**, –**s**, and –**N** options from the current system partition to the SP system.

–**n** Specifies that all nodes in a *noderange* are written. The range specification has syntax similar to that of frame or slot ranges. Nodes are numbered starting with 1, for frame 1 slot 1, up to the number of slots on the system (note that a node number can refer to an empty slot). A *noderange* can span frames (for example, 1–4,17–50) would refer to all nodes occupying slots 1–4 on frame 1 and 1–16 on frames 2 and 3, and slots 1 and 2 on frame 4.

–**w** Specifies a list of host names, separated by commas, to include in the working collective. Both this flag and the **a** flag can be included on the same command line. Duplicate host names are only included once in the working collective.

–**e** Specifies an exclusion list. Comma-separated host names specified are not written to standard output.

–**v** Specifies that only nodes that are responding according to the SDR have their host names written.

–**d** Specifies that IP addresses are returned as output.

–**l** Specifies that long host names be written. (This is lowercase **l**, as in **l**ist.)

–N Specifies a list of node groups. Each node group is resolved into nodes. The host names of these nodes are added to the host list. If **–G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

Operands

None.

Description

The **hostlist** command writes SP host names to standard output. The arguments to the command indicate the host names to be written. More than one flag can be specified, in which case, the hosts indicated by all the flags are written.

If no arguments are specified, **hostlist** writes the contents of a file specified by the WCOLL environment variable. If the WCOLL environment variable does not exist, the MP_HOSTFILE environment variable is used as the name of a POE host file to use for input. Finally, **./host.list** is tried. If none of these steps are successful, an error has occurred. The input file is in **dsh**-working-collective-file or POE-host-list-file format. Node pool specifications in POE host files are not supported.

Files

working collective file
See the **dsh** command.

POE host.list file
See *Parallel Environment for AIX: Operation and Use* documentation.

Related Information

Commands: **dsh**, **sysctl**

Examples

1. To create a working collective file of all nodes in the system partition that are responding, except for badhost, enter:

   ```
   hostlist -av -e badhost > ./working
   ```

2. To run a program on the nodes on slot 1 of each of 4 frames, enter:

   ```
   hostlist -s 1-4:1 | dsh -w - program
   ```

3. To run a program on the nodes on all slots for frame 1 and slots 1-3 for frame 3, as well as on host **otherone**, enter:

   ```
   hostlist -n 1-16,33-35 -w otherone | dsh -w - program
   ```

4. To run a Sysctl application on all the nodes in the WCOLL file **./wcoll:**, enter:

   ```
   export WCOLL=./wcoll
   hostlist | sysctl -c - sysctl_app args
   ```

hr script

Purpose

hr – Controls the host_responds monitor daemon, **hrd**, on the control workstation.

Syntax

hr [–**spname** *syspar_name*]

{ [**start** | **resume**] | [**stop** | **quiesce**] | **reset** |

[**query** | **qall** | **qsrc**] | **refresh** | **mksrc** *optional_flags* | **rmsrc** | **clean** | **restore** |

[**debug** | **debug off**] | [**trace on** | **trace off**] }

Flags

–**spname** *syspar_name*
: Executes the command for the system partition specified by the *syspar_name* operand. If this flag is not specified, the name of the system partition given by the value of the **SP_NAME** variable is used.

Operands

start | **resume**
: Starts the **hrd** daemon.

stop | **quiesce**
: Stops the **hrd** daemon.

reset
: Stops and restarts the **hrd** daemon.

query
: Queries the daemon for status. The response to the **query** includes **hrd**-specific information.

qall
: Performs the **query** function for each defined partition.

qsrc
: Displays a subsystem definition for a partition.

refresh
: Uses the refresh command to request a daemon refresh.

mksrc *optional_flags*
: Uses the **mkssys** command to create an SRC subsystem object. Additional flags for the command may be specified.

rmsrc
: Uses the **rmssys** command to remove an SRC subsystem object.

clean
: Removes all entries for the subsystem for all system partitions.

restore
: Synchronizes the running daemons with the information in the System Data Repository (SDR). This operand removes all entries for the subsystem, creates new entries based on information in the SDR, and starts the subsystems.

[debug | **debug off]**
: Turns debugging on or off.

[trace on | **trace off]**
: Turns additional tracing on or off.

Description

Use this command to control the operation of **hrd**, the host_responds daemon on the control workstation within a system partition. The heartbeat server provides input to the host_responds function within a system partition for the System Monitor through the **hrd** daemons.

The **hr** script is not normally executed from the command line. It is normally called by the **hrctrl** command, which is in turn called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **hrd** daemon is initially started on the control workstation with the System Resource Controller (SRC). It is respawned automatically if the **hrd** daemon fails. The SP_NAME environment variable causes selection of the correct daemon.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

The “Starting Up and Shutting Down the SP System” chapter and “The System Data Repository” appendix in *IBM Parallel System Support Programs for AIX: Administration Guide*.

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hr

Related Information

Commands: **hb**, **hrctrl**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

See the **hrctrl** command.

hrctrl script

Purpose

hrctrl – A script that controls the Host_Responds subsystem.

Syntax

hrctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem.
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystems, that is, delete them from all system partitions.
–**t**	Turns tracing on for the subsystem.
–**o**	Turns tracing off for the subsystem.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

The Host_Responds subsystem provides to other PSSP subsystems information about the state of the nodes on the IBM RS/6000 Scalable POWERparallel Systems (SP).

The **hrctrl** control script controls the operation of the Host_Responds subsystem. The subsystem is under the control of the System Resource Controller (SRC) and belongs to a subsystem group called **hr**. Associated with each subsystem is a daemon and a script that configures and starts the daemon.

An instance of the Host_Responds subsystem executes on the control workstation for every system partition. Because Host_Responds provides its services within the scope of a system partition, its subsystem is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. The script should be issued on the control workstation. If it is issued on a node, it has no effect.

From an operational point of view, the Host_Responds subsystem group is organized as follows:

Subsystem Host_Responds

Subsystem Group **hr**

SRC Subsystem **hr**

The **hr** subsystem is associated with the **hrd** daemon and the **hr** script. The **hr** script configures and starts the **hrd** daemon.

On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **hr.sp_prod** and **hr.sp_test**.

The subsystem does not run on the nodes.

Daemons **hrd**

The **hrd** daemon provides the Host_Responds services. The **hr** script configures and starts the **hrd** daemon.

The **hrctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **hrctrl** script provides a variety of controls for operating the Host_Responds subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off
- Refreshing the subsystem.

Before performing any of these functions, the script obtains the node number (using the **node_number**) command. If the node number is not zero, the control script is running on a node and it exits immediately. Otherwise, it is executing on the control workstation and it calls the **hr** script with an operand that specifies the action to be performed.

Adding the Subsystem

When the –**a** flag is specified, the control script uses the **hr** command with the **mksrc** operand to add the Host_Responds subsystem to the SRC.

Starting the Subsystem

When the –**s** flag is specified, the control script uses the **hr** command with the **start** operand to start the Host_Responds subsystem, **hr**.

Stopping the Subsystem

When the –**k** flag is specified, the control script uses the **hr** command with the **stop** operand to stop the Host_Responds subsystem, **hr**.

Deleting the Subsystem

When the –**d** flag is specified, the control script uses the **hr** command with the **rmsrc** operand to remove the Host_Responds subsystem from the SRC.

Cleaning up the Subsystems

When the –**c** flag is specified, the control script uses the **hr** command with the **clean** operand to stop and remove the Host_Responds subsystems for all system partitions from the SRC.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **hrd** daemon, using the **hr** command with the **trace on** operand.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off (returns it to its default level) for the **hrd** daemon, using the **hr** command with the **trace off** operand.

Refreshing the Subsystem

When the –**r** flag is specified, the control script refreshes the subsystem, using the **hr refresh** command.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/hrctrl

Related Information

Commands: **hr**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

Examples

1. To add the Host_Responds subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -a
   ```

2. To start the Host_Responds subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -s
   ```

3. To stop the Host_Responds subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -k
   ```

4. To delete the Host_Responds subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -d
   ```

5. To clean up the Host_Responds subsystem on all system partitions, enter:

   ```
   hrctrl -c
   ```

6. To turn tracing on for the Host_Responds daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -t
   ```

7. To turn tracing off for the Host_Responds daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name and enter:

   ```
   hrctrl -o
   ```

8. To display the status of all of the subsystems in the Host_Responds SRC group, enter:

   ```
   lssrc -g hr
   ```

9. To display the status of an individual Host_Responds subsystem, enter:

   ```
   lssrc -s subsystem_name
   ```

10. To display detailed status about an individual Host_Responds subsystem, enter:

    ```
    lssrc -l -s subsystem_name
    ```

    In response, the system returns information that includes the running status of the subsystem and the status of the nodes within the system partition.

11. To display the status of all of the daemons under SRC control, enter:

    ```
    lssrc -a
    ```

hsdatalst

Purpose

hsdatalst – Displays data striping device (HSD) data for the IBM Virtual Shared Disks from the System Data Repository (SDR).

Syntax

hsdatalst [–**G**]

Flags

–**G** Displays information for all system partitions on the SP, not only the current system partition.

Operands

None.

Description

This command is used to display defined HSD information in the system.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit list_vsd
```

and select the List Defined Hashed Shared Disk option.

Files

/usr/lpp/csd/bin/hsdatalst Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defhsd**, **undefhsd**

Examples

To display SDR HSD data, enter:

```
hsdatalst
```

hsdvts

Purpose

hsdvts – Verifies that a data striping device (HSD) for the IBM Virtual Shared Disks has been correctly configured and works.

Syntax

hsdvts *hsd_name*

Flags

None.

Operands

hsd_name The name of the HSD you want verified. **Warning:** Data on *vsd_name* will be overwritten and, therefore, destroyed.

Description

Warning

Data on *hsd_name* will be overwritten and, therefore, destroyed. Use this command after you have defined your HSD, IBM Virtual Shared Disks, and logical volumes, but **before** you have loaded your application data onto any of them.

This command writes /**unix** to *hsd_name*, reads it from *hsd_name* to a temporary file, and compares the temporary file to the original to make sure the I/O was successful. If the files compare exactly, the test was successful.

hsdvts writes to the raw *hsd_name* device /**dev**/**r***hsd_name*. Since raw devices can only be written in multiples of 512-sized blocks, **hsdvts** determines the number of full 512-byte blocks in /**unix** file, and writes that number to *hsd_name* via **dd** command. It makes a copy of /**unix** that contains this number of 512-byte blocks for comparison to the copy read from *hsd_name*. The **dd** command is used for all copy operations.

Files

/**usr**/**lpp**/**csd**/**bin**/**hsdvts** Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsd**, **cfgvsd**, **dd**, **defhsd**, **startvsd**

ifconfig

Purpose

/usr/lpp/ssp/css/ifconfig – Configures or displays network interface parameters for a network using TCP/IP.

Syntax

ifconfig *interface* [*address_family* [*address* [*destination_address*]] [*parameter*...]]

Flags

None.

Operands

address Specifies the network address for the network interface. For the inet family, the *address* operand is either a host name, or an Internet address in the standard dotted decimal notation.

address_family Specifies which network address family to change. The inet and ns address families are currently supported. This operand defaults to the inet address family.

destination_address
Specifies the address of the correspondent on the remote end of a point-to-point link.

interface Specifies the network interface configuration values to show or change. You must specify an interface with the *interface* operand when you use the **ifconfig** command. Abbreviations for the interfaces include:

- **en** Standard Ethernet (inet, xns)
- **et** IEEE 802.3 Ethernet (inet, xns)
- **tr** Token ring (inet, xns)
- **xt** X.25 (inet)
- **sl** Serial line IP (inet)
- **lo** Loopback (inet)
- **op** Serial (inet)
- **css** Scalable POWERparallel Switch (SP Switch) or High Performance Switch

Include a numeral after the abbreviation to identify the specific interface (for example, tr0).

parameter Allows the following parameter values:

alias Establishes an additional network address for the interface. When changing network numbers, this is useful for accepting packets addressed to the old interface.

allcast Sets the token-ring interface to broadcast to all rings on the network.

–allcast Confines the token-ring interface to broadcast only to the local ring.

arp Enables the **ifconfig** command to use the Address Resolution Protocol (ARP) in mapping between network-level addresses and link-level addresses. This flag is in effect by default.

–arp Disables the use of the Address Resolution Protocol.

authority Reserved.

bridge Reserved.

–bridge Reserved.

broadcast_address
(inet only). Specifies the address to use to broadcast to the network. The default broadcast address has a host part of all 1's (ones).

debug Enables driver-dependent debug code.

–debug Disables driver-dependent debug code.

delete Removes the specified network address. This is used when an alias is incorrectly specified or when it is no longer needed. Incorrectly setting ns addresses have the side effect of specifying the host portion of the network address. Removing all ns addresses allows you to respecify the host portion.

detach Removes an interface from the network interface list. If the last interface is detached, the network interface driver code is unloaded.

down Marks an interface as inactive (down), which keeps the system from trying to transmit messages through that interface. If possible, the **ifconfig** command also resets the interface to disable reception of messages. Routes that use the interface, however, are not automatically disabled.

hwloop Enables hardware loopback. The hardware loopback specifies that locally-addressed packets handled by an interface should be sent out using the associated adapter.

–hwloop Disables hardware loopback. The hardware loopback specifies that locally-addressed packets handled by an interface should be sent out using the associated adapter.

ipdst Specifies an Internet host willing to receive IP packets encapsulating ns packets bound for a remote network. An apparent point-to-point link is constructed, and the specified address is taken as the ns address and network of the destination.

metric_number
Sets the routing metric of the interface to the value specified by the *number* variable. The default is 0.

The routing metric is used by the routing protocol (the routed daemon). Higher metrics have the effect of making a route less favorable. Metrics are counted as addition hops to the destination network or host.

mtu_value Sets the maximum IP packet size for this system. The *value* variable can be any number from 60 through 65520, depending on the network interface. See "Understanding Automatic Configuration of Network Interfaces" in *AIX Version 4 System Management Guide: Communications and Networks* for maximum transmission unit (MTU) values by interface.

netmask_mask
Specifies how much of the address to reserve for subdividing networks into subnetworks. This parameter can only be used with an address family of inet.

The *mask* variable includes both the network part of the local address and the subnet part, which is taken from the host field of the address. The *mask* can be specified as a single hexadecimal number beginning with 0x, in standard Internet dotted decimal notation, or beginning with a name or alias that is listed in the **/etc/networks** file.

The *mask* contains 1's (ones) for the bit positions in the 32-bit address that are reserved for the network and subnet parts, and 0's (zeros) for the bit positions that specify the host. The *mask* should contain at least the standard network portion, and the subnet segment should be contiguous with the network segment.

offset Used by the CSS/IP for static IP address translation only.

Note: If the ARP is enable, offset is not used.

TB0/TB2 Indicates to the CSS/IP whether it is running over TB0 or TB2 adapter interface. The default is TB2 adapter.

security Reserved.

snap Reserved.

–snap Reserved.

up Marks an interface as active (up). This parameter is used automatically when setting the first address for an interface. It can also be used to enable an interface after an **ifconfig down** command.

Description

The **ifconfig** command has been modified to add support for the switch. This command is valid only on an SP system.

The **ifconfig** command can be used from the command line either to assign an address to a network interface, or to configure or display the current network interface configuration information. The **ifconfig** command must be used at system start up to define the network address of each interface present on a machine. It can also be used at a later time to redefine an interface's address or other operating parameters. The network interface configuration is held on the running system and must be reset at each system restart.

An *interface* can receive transmissions in differing protocols, each of which may require separate naming schemes. It is necessary to specify the *address_family* parameter, which can change the interpretation of the remaining parameters. The address families currently supported are inet and ns.

For the DARPA Internet family, inet, the address is either a host name present in the host name database, that is, the /**etc**/**hosts** file, or a DARPA Internet address expressed in the Internet standard dotted decimal notation.

For the Xerox Network Systems (XNS) family, ns, addresses are net:a.b.c.d.e.f., where net is the assigned network number (in decimal), and each of the six bytes of the host number, a through f, are specified in hexadecimal. The host number can be omitted on 10-Mbps Ethernet interfaces, which use the hardware physical address, and on interfaces other than the first interface.

While any user can query the status of a network interface, only a user who has administrative authority can modify the configuration of those interfaces.

Related Information

AIX Command: **netstat**

AIX Files: /**etc**/**host**, /**etc**/**networks**

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the SP Switch and the High Performance Switch.

Refer to *AIX Version 4 System Management Guide: Communications and Networks* for additional information on TCP/IP protocols.

Refer to *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for an overview on Xerox Network Systems (XNS).

Examples

The following are examples using the **ifconfig** command on a TCP/IP network and an XNS network, respectively:

Inet Examples

1. To query the status of a serial line IP interface, enter:

```
ifconfig sl1
```

In this example, the interface to be queried is sl1. The result of the command looks similar to the following:

```
sl1: flags=51<UP,POINTOPOINT,RUNNING>
     inet 192.9.201.3 --> 192.9.354.7 netmask ffffff00
```

2. To configure the local loopback interface, enter:

   ```
   ifconfig lo0 inet 127.0.0.1 up
   ```

3. To mark the local token-ring interface as down, enter:

   ```
   ifconfig tr0 inet down
   ```

   In this example, the interface to be marked is token0.

   Note: Only a user with root user authority can modify the configuration of a network interface.

4. To specify an alias, enter:

   ```
   ifconfig css0 inet 127.0.0.1 netmask 255.255.255.0 alias
   ```

XNS Examples

1. To configure a standard Ethernet-type interface for XNS, enter:

   ```
   ifconfig en0 ns 110:02.60.8c.2c.a4.98 up
   ```

   In this example, ns is the XNS address family, 110 is the network number and 02.60.8c.2c.a4.98 is the host number, which is the Ethernet address unique to each individual interface. Specify the host number when there are multiple Ethernet hardware interfaces, as the default may not correspond to the proper interface. The Ethernet address can be obtained by the commands:

   ```
   ifconfig en0
   netstat -v
   ```

   The XNS address can be represented by several means, as can be seen in the following examples:

   ```
   123#9.89.3c.90.45.56

   5-124#123-456-900-455-749

   0x45:0x9893c9045569:90

   0456:9893c9045569H
   ```

   The first example is in decimal format, and the second example, using minus signs, is separated into groups of three digits each. The 0x and H examples are in hexadecimal format. Finally, the 0 in front of the last example indicates that the number is in octal format.

2. To configure an IEEE Ethernet 802.3-type interface for XNS, enter:

   ```
   ifconfig et0 ns 120:02.60.8c.2c.a4.98 up
   ```

   The en0 and et0 interfaces are considered as separate interfaces even though the same Ethernet adapter is used. Two separate networks can be defined and used at the same time as long as they have separate network numbers. Multiple Ethernet adapters are supported.

   Note: The host number should correspond to the Ethernet address on the hardware adapter. A system can have multiple host numbers.

3. To configure an Internet encapsulation XNS interface, enter:

```
ifconfig en0 inet 11.0.0.1 up
ifconfig en0 ns 110:02.60.8c.2c.a4.98 up
ifconfig en0 ns 130:02.60.8c.34.56.78 ipdst 11.0.0.10
```

The first command brings up the Internet with the inet address 11.0.0.1. The second command configures the en0 interface to be network 110 and host number 02.60.8c.2c.a4.98 in the ns address family. This defines the host number for use when the XNS packet is encapsulated within the Internet packet. The last command defines network 130, host number 02.60.8c.34.56.78, and destination Internet address 11.0.0.10. This last entry creates a new network interface, nsip. Use the **netstat** –**i** command for information about this interface.

install_cw

Purpose

install_cw – Completes the installation of system support programs in the control workstation.

Syntax

install_cw

Flags

None.

Operands

None.

Description

Use this command at installation to perform the following tasks:

- Installs IBM Parallel System Support Programs for AIX (PSSP) SMIT Panels
- Starts and configures the System Data Repository (SDR)
- Sets the node number to 0 in the Object Data Management (ODM) table
- Starts and configures PSSP daemons

install_hacws

Purpose

install_hacws – Creates and configures a High Availability Control Workstation (HACWS) configuration from a regular control workstation configuration.

Syntax

install_hacws –**p** *host_name* –**b** *host_name* [–**s**]

Flags

–**p** Specifies the host name of the primary control workstation. The host name is the name that is set in the kernel and identifies the physical machine. It is also required that this name have a route defined to a network adapter on the primary control workstation. This option is required.

–**b** Specifies the host name of the backup control workstation. The host name is the name that is set in the kernel and identifies the physical machine. It is also required that this name have a route defined to a network adapter on the backup control workstation. This option is required.

–**s** Invokes the command on both the primary and the backup control workstations.

Operands

None.

Description

Use this command to perform configuration and installation tasks on HACWS. This command is used instead of **install_cw** once the configuration has been made an HACWS configuration. This command is valid only when issued on the control workstation. When the command is executed and the calling process is not on a control workstation, an error occurs.

Note: The **install_hacws** command permanently alters a control workstation to an HACWS. The only way to go back to a single control workstation is to have a **mksysb** image of the primary control workstation before the **install_hacws** command is executed.

Both the primary and backup control workstations must be running and capable of executing remote commands via the **/usr/lpp/ssp/rcmd/bin/rsh** command.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred. Diagnostic information is written to standard output and standard error.

Standard output consists of messages indicating the progress of the command as it configures the control workstations.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on the HACWS option.

Location

/usr/sbin/hacws/install_hacws

Related Information

SP Commands: **install_cw**, **rsh**, **setup_logd**

Examples

1. To configure both control workstations on an SP system, enter the following:

   ```
   install_hacws -p primary_cw -b backup_cw -s
   ```

2. To configure the control workstations separately, enter the following.

   On the primary control workstation, enter:

   ```
   install_hacws -p primary_cw -b backup_cw
   ```

   After the preceding command completes on the primary control workstation, enter the following on the backup control workstation:

   ```
   install_hacws -p primary_cw -b backup_cw
   ```

jm_config

Purpose

jm_config – Reconfigures the Resource Manager.

Syntax

jm_config

Flags

None.

Operands

None.

Description

Use this command to reconfigure the Resource Manager (RM) servers.

This command must be executed by root on the control workstation. It reads the Resource Manager configuration data from the /**etc/jmd_config.***syspar_name* file, where *syspar_name* represents the current system partition environment. This current working environment can be determined by issuing **spget_syspar** –**n**. The **jm_config** command then contacts the correct primary Resource Manager and sends a message telling the server to update its configuration data from the System Data Repository (SDR). The new configuration takes effect with the next client request.

Note: 604 High Nodes cannot be configured as part of a parallel pool. Therefore, the Resource Manager will not allocate these nodes for parallel jobs.

The Resource Manager can also be reconfigured via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit RM_options
```

and select the Reconfigure the Resource Manager option. Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on configuring the Resource Manager and system partitioning.

Files

/etc/jmd_config.*syspar_name*
Resource Manager configuration data file.

/usr/lpp/ssp/bin/jm_config
Path name of this command.

/var/adm/SPlogs/jm/jmd_out
Resource Manager information log.

/var/adm/SPlogs/jm/jmd_err
Resource Manager error log.

Related Information

Commands: **jm_start**, **jm_status**, **jm_stop**, **locate_jm**, **spget_syspar**

Examples

To reconfigure the Resource Manager in the current system partition, enter:

```
jm_config
```

The following response shows the configuration data file being used by the **jm_config** command:

```
jm_config: Using /etc/jmd_config.k42sp1 for Resource Manager
           configuration data.
jm_config: jm_config successful
```

jm_install_verify

Purpose

jm_install_verify – Verifies that the installation of the Resource Manager component of the SP system completed successfully.

Syntax

jm_install_verify [–**h**] [–**q**] [–**l** *logfile*]

Flags

–**h**	Displays usage information.
–**q**	Specifies quiet mode; suppresses output to standard output.
–**l** *logfile*	Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

Operands

None.

Description

Use this command to perform various tests to determine whether the Resource Manager component of the SP system is installed correctly. This command checks that:

- The directories, files, and links are correct
- The System Data Repository (SDR) classes used by the Resource Manager exist
- The entries made in the /**etc**/**services** and /**etc**/**inetd.conf** files are correct

This command must be executed by root and operates within the scope of the current system partition environment. When executed on the control workstation, installation will be verified for the control workstation and all of the nodes within the current system partition. When executed on a node, installation is verified on that node only.

If you do not specify the –**l** flag, more detailed information is recorded in /**var**/**adm**/**SPlogs**/**jm_install_verify.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the Resource Manager Installation option.

Files

/**usr**/**lpp**/**ssp**/**bin**/**jm_install_verify**
Path name of this command.

/**var**/**adm**/**SPlogs**/**jm_install_verify.log**
Default log file.

Exit Values

0 Indicates success.

1 Indicates a failure with an appropriate message.

If you do not specify the –**q** flag, a message is displayed on standard output indicating the success or failure of the tests. If errors are detected, a message is displayed stating how many errors were found and more detailed information is recorded in the log file.

Related Information

Commands: **CSS_test**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_ctest**, **spmon_itest**

Examples

To verify installation of the Resource Manager on a single node, saving error messages in the **jminst.err** file, from that node, enter:

```
jm_install_verify -l jminst.err
```

Upon successful completion, the following message is displayed:

```
Verifying installation of Resource Manager on node 9.

Resource Manager installation verification SUCCESSFUL on node 9.

Check ./jminst.err file for details.
```

jm_start

Purpose

jm_start – Starts the primary and backup Resource Manager servers in the current system partition.

Syntax

jm_start [–**h** | –**i**]

Flags

–**h** Displays usage information.

–**i** Starts the Resource Manager without determining whether the server is already running.

Operands

None.

Description

Use this command to start the primary and backup Resource Manager servers.

This command must be executed by root on the control workstation. It reads the Resource Manager configuration data from the /**etc/jmd_config.***syspar_name* file, where *syspar_name* represents the current system partition environment. This current working environment can be determined by issuing **spget_syspar** –**n**. This command reads the configuration data into the System Data Repository (SDR) and starts the primary Resource Manager server on one of the nodes indicated. This server then starts up a backup on the next available node in the list of server candidates.

Note: 604 High Nodes cannot be configured as part of a parallel pool. Therefore, the Resource Manager will not allocate these nodes for parallel jobs.

If **jm_start** successfully starts the primary Resource Manager server, it returns a message indicating its location. If a backup is not successfully started, an additional message is issued and the primary Resource Manager server continues to run, attempting periodically to start the backup. If you do not have a backup running, and the primary Resource Manager server dies for any reason, there is no recovery and the Resource Manager server must be restarted manually.

Use the –**i** flag only if the **jm_start** command cannot determine whether the Resource Manager server is already running. The –**i** flag does not check if the Resource Manager server is already running, therefore, it is best not to use this option unless you are sure the Resource Manager server is not running on any node.

The Resource Manager can also be reconfigured via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit RM_options
```

and select the Start the Resource Manager option. Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the Resource Manager and system partitioning.

Files

/etc/jmd_config.*syspar_name*
Resource Manager configuration data file.

/usr/lpp/ssp/bin/jm_start
Path name of this command.

/var/adm/SPlogs/jm/jmd_out
Resource Manager information log.

/var/adm/SPlogs/jm/jmd_err
Resource Manager error log.

Related Information

Commands: **jm_config**, **jm_status**, **jm_stop**, **locate_jm**, **spget_syspar**

Examples

To start the Resource Manager in the current system partition, enter:

```
jm_start
```

The following response shows the configuration data file being used by the **jm_start** command:

```
jm_start: Using /etc/jmd_config.k42sp1 for Resource Manager
          configuration data.
jm_start: jmd started on k42n09:
```

jm_status

Purpose

jm_status – Gets information about defined pools or about jobs running on nodes allocated by the Resource Manager.

Syntax

jm_status [–**n** *control_workstation_name*]
{–**P** | –**p** *pool_id* | –**j**} [–**h**]
[–**v**]

Flags

–**n***control_workstation_name*
Indicates the name or IP alias name of the control workstation or an SP system partition on the control workstation. This flag is optional. If you choose not to specify the –**n** flag, you can set the SP_NAME environment variable to indicate the current system partition environment. If neither is specified, the current environment is obtained from the local /**etc/SDR_dest_info** file.

–**P** Requests information about all the pools in the SP system.

–**p** *pool_id* Requests information about the pool specified by *pool_id*. A *pool_id* value of –2 requests information about all the parallel pools. A *pool_id* value of –3 requests information about all pools and is equivalent to specifying –**P**.

–**j** Requests information about all the jobs running on the system.

–**h** Displays usage information.

–**v** The –**v** (verbose) flag is always accepted, but no special action is taken when it is specified together with –**j**. If –**P** or –**p** is specified:

- Without the –**v** flag, only information about pool IDs, pool attributes and member nodes is displayed.
- With the –**v** flag, additional information is displayed about jobs running in the specified pools.

Operands

None.

Description

Use the **jm_status** command to get information about defined pools or about jobs running on nodes allocated by the Resource Manager. Any user can execute **jm_status** on any workstation.

Files

/usr/lpp/ssp/bin/jm_status
Path name of this command

/etc/SDR_dest_info
System Data Repository information file.

Related Information

Commands: **locate_jm**, **spget_syspar**

Examples

1. To get job information for the k42sp1 partition, enter:

   ```
   jm_status -n k42sp1 -j
   ```

   The response should be similar to this:

   ```
   Job 1:    time_allocated=Mon_Apr__4_16:09:28_1994
             description=interpret_alloc_imp_test
             requestor=root requestor_pid=11787
             requestor_node=r07n03.hpssl.kgn.ibm.com
             Adapter type=HPS_US
       Node: r07n01.hpssl.kgn.ibm.com
             Usage: cpu=SHARED adapter=DEDICATED
             virtual task ids: 0 1
       Node: r07n09.hpssl.kgn.ibm.com
             Usage: cpu=SHARED adapter=DEDICATED
             virtual task ids: 2 3 4
   Job 2:    time_allocated=Mon_Apr__4_16:13:08_1994
             description=interpret_alloc_imp_test
             requestor=root requestor_pid=12355
             requestor_node=r07n03.hpssl.kgn.ibm.com
             Adapter type=ETHERNET
       Node: r07n01.hpssl.kgn.ibm.com
             Usage: cpu=SHARED adapter=DEDICATED
             virtual task ids: 0 1
       Node: r07n09.hpssl.kgn.ibm.com
             Usage: cpu=SHARED adapter=DEDICATED
             virtual task ids: 2 3 4
   Job 3:    time_allocated=Mon_Apr__4_16:13:38_1994
             description=interpret_alloc_imp_test
             requestor=root requestor_pid=14404
             requestor_node=r07n03.hpssl.kgn.ibm.com
             Adapter type=HPS_US
       Node: r07n15.hpssl.kgn.ibm.com
             Usage: cpu=SHARED adapter=SHARED
             virtual task ids: 0 1
   ```

2. To get Pool information for the currently defined system partition environment, enter:

   ```
   jm_status -P
   ```

   The response should be similar to this:

```
Pool 1:    Ethernet_test_pool_1
  Subpool: INTERACTIVE
    Node:  r07n01.hpssl.kgn.ibm.com
  Subpool: BATCH
    Node:  r07n03.hpssl.kgn.ibm.com
    Node:  r07n13.hpssl.kgn.ibm.com
    Node:  r07n11.hpssl.kgn.ibm.com
  Subpool: GENERAL
    Node:  r07n15.hpssl.kgn.ibm.com
Pool 2:    Ethernet_test_pool_2
  Subpool: INTERACTIVE
    Node:  r07n09.hpssl.kgn.ibm.com
  Subpool: BATCH
    Node:  r07n05.hpssl.kgn.ibm.com
  Subpool: GENERAL
    Node:  r07n07.hpssl.kgn.ibm.com
```

3. Entering:

   ```
   jm_status -p1 -v
   ```

   should get a response similar to this:

   ```
   Pool 1:    Ethernet_test_pool_1
     Subpool: INTERACTIVE
       Node:  r07n01.hpssl.kgn.ibm.com
         Job 1: time allocated=Mon_Apr__4_16:09:28_1994
           description=interpret_alloc_imp_test
           requestor=root requestor pid=11787
           requestor_node=r07n03.hpssl.kgn.ibm.com
           Adapter type=HPS_US
           Usage: cpu=SHARED adapter=DEDICATED
           virtual task ids: 0 1
         Job 2: time allocated=Mon_Apr__4_16:13:08_1994
           description=interpret_alloc_imp_test
           requestor=root requestor pid=12355
           requestor_node=r07n03.hpssl.kgn.ibm.com
           Adapter type=ETHERNET
           Usage: cpu=SHARED adapter=DEDICATED
           virtual task ids: 0 1
     Subpool: BATCH
       Node:  r07n03.hpssl.kgn.ibm.com
       Node:  r07n13.hpssl.kgn.ibm.com
       Node:  r07n11.hpssl.kgn.ibm.com
     Subpool: GENERAL
       Node:  r07n15.hpssl.kgn.ibm.com
         Job 3: time allocated=Mon_Apr__4_16:13:38_1994
           description=interpret_alloc_imp_test
           requestor=root requestor pid=14404
           requestor_node=r07n03.hpssl.kgn.ibm.com
           Adapter type=HPS US
           Usage: cpu=SHARED adapter=SHARED
           virtual task ids: 0 1
   ```

jm_stop

Purpose

jm_stop – Stops the primary and backup Resource Manager servers in the current system partition.

Syntax

jm_stop [**–q** | **–h**]

Flags

–q Tells the Resource Manager to wait for existing jobs to terminate before exiting. If it is not specified, the primary and backup Resource Manager servers exit immediately.

–h Displays usage information.

Operands

None.

Description

Use this command to stop the primary and backup Resource Manager servers.

This command must be executed by root and operates within the scope of the current system partition environment. It can be issued from any node within a system partition.

The system can be quiesced in a less disruptive fashion by specifying the **–q** option, which tells the Resource Manager server to first wait for all existing jobs to terminate before exiting. The **jm_stop –q** command returns immediately. Any subsequent Resource Manager job requests, as well as **jm_status** and **jm_config** commands, fail. When all jobs have finished, the primary and backup Resource Manager server exits. Issue **locate_jm** to determine if the Resource Manager servers have exited. Anytime after **jm_stop –q** has been issued, you can issue **jm_stop** (without **–q**) to force an immediate stop.

The Resource Manager can also be stopped via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit RM_stop
```

and select either the Stop the Resource Manager immediately option (**jm_stop**), or the Wait for existing clients to exit, then stop the Resource Manager option (**jm_stop –q**).

Files

/usr/lpp/ssp/bin/jm_stop
Path name of this command.

Related Information

Commands: **jm_config**, **jm_start**, **jm_status**, **locate_jm**

Examples

Entering

```
jm_stop
```

should get this response:

```
jm_stop: jm_stop successful.
```

jm_verify

Purpose

jm_verify – Verifies that the configuration of the Resource Manager component of the SP system completed successfully in the current system partition.

Syntax

jm_verify [–**h**] [–**q**] [–**l** *logfile*]

Flags

–**h**	Displays usage information.
–**q**	Specifies quiet mode; suppresses output to standard output.
–**l** *logfile*	Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

Operands

None.

Description

Use this command to perform various tests to determine whether the Resource Manager component of the SP system is correctly configured.

This command can be executed by any user and operates within the scope of the current system partition environment.

This command utilizes the SP **dsh** and **rsh** commands to access remote nodes for configuration information. The same security considerations apply.

If you do not specify the –**l** flag, more detailed information is recorded in **/var/adm/SPlogs/jm_verify.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the Resource Manager Installation option.

Files

/usr/lpp/ssp/bin/jm_verify
Path name of this command.

/var/adm/SPlogs/jm_verify.log
Default log file.

Exit Values

0 Indicates success.

1 Indicates a failure with an appropriate message.

If you do not specify the –**q** flag, a message is displayed on standard output indicating the success or failure of the tests. If errors are detected, a message is displayed stating how many errors were found and more detailed information is recorded in the log file.

Related Information

Commands: **CSS_test**, **dsh**, **jm_install_verify**, **rsh**, **SDR_test**, **SYSMAN_test**, **spmon_ctest**, **spmon_itest**

Examples

To verify configuration of the Resource Manager, saving error messages in **jmconf.err** in the current working directory, enter:

```
jm_verify -l jmconf.err
```

Upon successful completion, the following message is displayed:

```
Verifying Resource Manager on node 0.

Resource Manager verification SUCCESSFUL on node 0.

Check ./jmconf.err file for details.
```

jmcmi_accesscontrol

Purpose

jmcmi_accesscontrol – Updates the *access_control* attribute of the JM_domain_info System Data Repository (SDR) class.

Syntax

jmcmi_accesscontrol [–**h**] *access_control {***yes** | **no***} update_config* {**yes** | **no**}

Flags

–**h** Displays usage information.

Operands

access_control
Specifies the value to set the *access_control* attribute to. Valid options are **yes** or **no**.

update_config
Specifies whether to notify the Resource Manager (RM) server when configuration changes are made. Valid options are **yes** or **no**.

Description

This command is a Perl script used to set the attribute value of *access_control* within the JM_domain_info SDR class. The *access_control* attribute determines whether the RM should interact with the system management command, **spacs_cntrl**, to allow users access to parallel nodes. If **yes** is specified for the *update_config* operand, the RM server is notified of the change to *access_control* and the new value is used after the next client connection. If **no** is specified, a second action must be taken before the new value is used. If the RM server is not running, the action must be to start the RM server. If the RM server is running, the action would be to reconfigure the RM server.

Executing this script is equivalent to setting the keyword ACCESS_CONTROL within the RM /**etc/jmd_config.***syspar_name* configuration file. The data is updated within the SDR and the /**etc/jmd_config.***syspar_name* file is updated with the new value for ACCESS_CONTROL.

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment. Both operands are required.

The main use of this script is via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit access_control
```

Files

/**etc/jmd_config.***syspar_name*
Resource Manager configuration data file.

Related Information

Commands: **jm_config**, **jm_start**, **jmcmi_createjmdconfig**, **spacs_cntrl**

Examples

To turn access control on and inform the RM server of the changes, enter:

```
jmcmi_accesscontrol yes yes
```

jmcmi_addpool

Purpose

jmcmi_addpool – Defines a new Resource Manager (RM) pool.

Syntax

jmcmi_addpool [–**h**] *pool_id attribute delete_nodes* {**yes** | **no**}
batch "*node_list*" **interactive** "*node_list*"
general "*node_list*" *update_config* {**yes** | **no**}

Flags

–**h** Displays usage information.

Operands

pool_id Specifies a unique pool ID. A –**1** must be used when defining the Serial pool and a nonnegative number must be used for parallel pools. It correlates to the POOL_ID keyword found within /**etc/jmd_config.***syspar_name* file.

attribute Specifies a descriptive string used to describe this RM pool. The string cannot contain blanks or an equals (=) sign. It correlates to the ATTRIBUTES keyword found within /**etc/jmd_config.***syspar_name* file.

delete_nodes
Specifies whether nodes defined within the *node_list* should be deleted from a previously defined pool and placed into this new pool. Valid options are **yes** or **no**.

"*node_list***"** Contains a list of node names to be defined for the subpool specified by the preceding keyword. The **batch** "*node_list*", **interactive** "*node_list*", and **general** "*node_list*" correlate to the MEMBERS_BATCH, MEMBERS_INTERACTIVE, and MEMBERS_GENERAL keywords found within the /**etc/jmd_config.***syspar_name* file.

update_config
Specifies whether the RM server is notified when configuration changes are made. Valid options are **yes** or **no**.

Description

This command is a Perl script used to define a new RM pool. For more information on RM pools and their definitions, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*.

After initial RM configuration, the RM Server List must be defined before the definition of any pools.

node_list must be one of the following:

- A list of node names that have a corresponding address and node number within the SDR

- The keyword **all** which takes all of the nodes listed within the SDR class, Node, and uses them as the *node_list*
- The keyword **none** which indicates that no nodes are to be defined for this subpool.

The keyword **all** when specified for a subpool works in combination with a *node_list* defined for another subpool, to create a subpool with all the nodes on the system excluding the ones specified by the *node_list*. See the Examples section for this command.

After this script is executed, the new pool data is updated within the System Data Repository (SDR) and the /**etc/jmd_config.***syspar_name* file is updated to reflect the new pool information.

If the *update_config* operand is **yes**, the RM server is notified that a configuration change was made, and it updates all of its configuration data. This data is used at the next client connection. If **no** is specified, a second action must be taken before the new configuration data is used. If the RM server is not running, the action must be to start the RM server. If the RM server is running, the action would be to reconfigure the RM server.

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment. All operands are required.

The main use of this script is via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit add_pool
```

Files

/**etc/jmd_config.***syspar_name*
Resource Manager configuration data file.

Related Information

Commands: **jm_config**, **jm_start**, **jmcmi_createjmdconfig**, **jmcmi_servernodes**

Examples

1. To create a new pool containing 8 batch nodes and 8 general nodes and inform the RM server of these changes, enter:

   ```
   jmcmi_addpool 1 pool_for_LL
   yes batch "r07n01 r07n02 r07n03 r07n04 r07n05
   r07n06 r07n07 r07n08" interactive "none" general
   "r07n09 r07n10 r07n11 r07n12 r07n13 r07n14 r07n15
   r07n16" yes
   ```

   This results in the following definition within the /**etc/jmd_config.***syspar_name* file:

```
POOL_ID = 1
ATTRIBUTES = pool_for_LL
MEMBERS_BATCH       = r07n01; r07n02; \
r07n03; r07n04; r07n05; \
r07n06; r07n07; r07n08;
MEMBERS_INTERACTIVE =
MEMBERS_GENERAL     = r07n09; \
r07n10; r07n11;r07n12;r07n13; \
r07n14;r07n15;r07n16;
```

If the node r07n10 was previously defined within Pool 2, **jmcmi_addpool** deletes r07n10 from that pool definition. The RM is notified of the new pool definitions and the configuration changes take effect on the next client connection.

2. To create a single pool with all of the nodes on the system within the general subpool and inform the RM server of these changes, enter:

   ```
   jmcmi_addpool 2 General_Pool
   yes batch "none" interactive "none" general
   "all" yes
   ```

   This results in all of the nodes being defined within Pool 2. Because the *delete_nodes* operand was specified as **yes**, any other pools that were defined on the system no longer exist. All of the nodes were deleted from the pools and moved to the one being created. The RM is notified of the new pool definitions and the configuration changes take effect at the next client connection.

3. To create a single pool with all of the nodes on the system, defining two nodes for the interactive subpool, two nodes for the batch subpool, and the remaining nodes for the general subpool, and inform the RM server of these changes, enter:

   ```
   jmcmi_addpool 2 DeptXYZ
   yes batch "r07n01 r07n16" interactive "r07n02 r07n15"
   general "all" yes
   ```

   This results in the following definition within the /**etc/jmd_config.***syspar_name* file:

   ```
   POOL_ID = 2
   ATTRIBUTES = DeptXYZ
   MEMBERS_BATCH       = r07n01; r07n16;
   MEMBERS_INTERACTIVE = r07n02; r07n15;
   MEMBERS_GENERAL     = r07n03; \
   r07n04; r07n05; \
   r07n06; r07n07; r07n08; \
   r07n09; r07n10; r07n11; \
   r07n12;r07n13; r07n14;
   ```

jmcmi_changepool

Purpose

jmcmi_changepool – Changes the definition of a Resource Manager (RM) pool.

Syntax

jmcmi_changepool [–**h**] *pool_id attribute delete_nodes* {**yes** | **no**}
batch "*node_list*" **interactive** "*node_list*"
general "*node_list*" *update_config* {**yes** | **no**}

Flags

–**h** Displays usage information.

Operands

pool_id Specifies the existing pool ID. It correlates to the POOL_ID keyword found within /**etc**/**jmd_config.***syspar_name*.

attribute Specifies a descriptive string used to describe this RM pool. The string cannot contain blanks or an equal (=) sign. It correlates to the ATTRIBUTES keyword found within /**etc**/**jmd_config.***syspar_name*.

delete_nodes
Specifies whether nodes defined within the *nodes_list* should be deleted from a previously defined pool and placed into this new pool. Valid options are **yes** or **no**.

"*node_list***"** Contains a list of node names to be defined for the subpool specified by the preceding keyword. The **batch** "*node_list*", **interactive** "*node_list*", and **general** "*node_list*" correlate to the MEMBERS_BATCH, MEMBERS_INTERACTIVE, and MEMBERS_GENERAL keywords found within the /**etc**/**jmd_config.***syspar_name* file.

update_config
Specifies whether the RM server is notified when configuration changes are made. Valid options are **yes** or **no**.

Description

This command is a Perl script used to redefine a RM pool. For more information on Resource Manager pools and their definitions, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*.

The *pool_id* operand cannot be changed by this script. In order to change a pool ID, the original pool must be deleted and a new one created.

"*node_list*" must be one of the following:

- A list of node names that has a corresponding address and node number within the System Data Repository (SDR)
- The keyword **all** which takes all of the nodes listed within the SDR class, node, and uses them as the "*node_list*"
- The keyword **none** which indicates no nodes are to be defined for this subpool

The keyword **all** when specified for a subpool works in combination with a "*node_list*" defined for another subpool, to create a subpool with all the nodes on the system excluding the ones specified by the "*node_list*." See the Examples section.

After this script is executed, the changed pool data is updated within the SDR and the /**etc/jmd_config.***syspar_name* file is updated to reflect the information.

If the *update_config* operand is **yes**, the RM server is notified that a configuration change was made, and updates all of its configuration data. This data is used at the next client connection. If **no** is specified, a second action must be taken before the new configuration data is used. If the RM server is not running, the action must be to start the RM server. If the RM server is running, the action would be to reconfigure the RM server.

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment. All operands are required.

The main use of this script is via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit configure_pool
```

Files

/**etc/jmd_config.***syspar_name*
: Resource Manager configuration data file.

Related Information

Commands: **jm_config**, **jm_start jmcmi_addpool**, **jmcmi_createjmdconfig**, **jmcmi_deletepool**

Examples

1. To redefine Pool 1 so that it contains 8 batch nodes and 8 general nodes, and inform the RM server of these changes, enter:

   ```
   jmcmi_changepool 1 pool_for_LL
   yes batch "r07n01 r07n02 r07n03 r07n04 r07n05
   r07n06 r07n07 r07n08" interactive "none" general
   "r07n09 r07n10 r07n11 r07n12 r07n13 r07n14 r07n15
   r07n16" yes
   ```

   This results in the following definition within the /**etc/jmd_config.***syspar_name* file:

   ```
   POOL_ID = 1
   ATTRIBUTES = pool_for_LL
   MEMBERS_BATCH       = r07n01; r07n02; \
   r07n03; r07n04; r07n05; \
   r07n06; r07n07; r07n08;
   MEMBERS_INTERACTIVE =
   MEMBERS_GENERAL     = r07n09; \
   r07n10; r07n11;r07n12;r07n13; \
   r07n14;r07n15;r07n16;
   ```

If the node r07n10 was previously defined within Pool 2, **jmcmi_changepool** deletes r07n10 from that pool definition. The RM is notified of the new pool definitions and the configuration changes take effect on the next client connection.

2. To redefine a single pool with all of the nodes on the system within the general subpool, enter:

```
jmcmi_changepool 2 General_Pool
yes batch "none" interactive "none" general
"all" yes
```

This results in all of the nodes being defined within Pool 2. Because the **delete_nodes** operand was specified as **yes**, any other pools that were defined on the system no longer exist. All of the nodes were deleted from the pools and moved to the one being redefined. The RM is notified of the new pool definitions and the configuration changes take effect at the next client connection.

3. To redefine a single pool with all of the nodes on the system, defining two nodes for the interactive subpool, two nodes for the batch subpool, and the remaining nodes for the general subpool, and inform the RM server of these changes, enter:

```
jmcmi_changepool 2 DeptXYZ
yes batch "r07n01 r07n16" interactive "r07n02 r07n15"
general "all" yes
```

This results in the following definition within the /**etc**/**jmd_config.***syspar_name* file:

```
POOL_ID = 2
ATTRIBUTES = DeptXYZ
MEMBERS_BATCH       = r07n01; r07n16;
MEMBERS_INTERACTIVE = r07n02; r07n15;
MEMBERS_GENERAL     = r07n03; \
r07n04; r07n05; \
r07n06; r07n07; r07n08; \
r07n09; r07n10; r07n11; \
r07n12;r07n13; r07n14;
```

jmcmi_createjmdconfig

Purpose

jmcmi_createjmdconfig – Updates the Resource Manager (RM) configuration file, /**etc**/**jmd_config.***syspar_name*, with the current configuration data.

Syntax

jmcmi_createjmdconfig

Flags

None.

Operands

None.

Description

This Perl script is executed by other RM configuration scripts to update the /**etc**/**jmd_config.***syspar_name* file with any changes made during their execution.

The existing /**etc**/**jmd_config.***syspar_name* is preserved as a unique file name /**etc**/**jmd_config.***syspar_name***.{***timestamp***}**, so that no previous configuration data is lost. A new /**etc**/**jmd_config.***syspar_name* file is created by getting the current configuration data from the System Data Repository (SDR).

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment.

Files

/**etc**/**jmd_config.***syspar_name*
: Resource Manager configuration data file.

Related Information

Commands: **jmcmi_accesscontrol**, **jmcmi_addpool**, **jmcmi_changepool**, **jmcmi_deletepool**, **jmcmi_servernodes**

jmcmi_deletepool

Purpose

jmcmi_deletepool – Deletes a pool definition from the Resource Manager (RM).

Syntax

jmcmi_deletepool [–**h**] *pool_id update_config* {**yes** | **no**}

Flags

–**h** Displays usage information.

Operands

pool_id Specifies the existing pool ID of the pool for which the data is deleted. It correlates to the POOL_ID keyword found within the /**etc/jmd_config.***syspar_name* file.

update_config
Specifies whether the RM server is notified when configuration changes are made. Valid options are **yes** or **no**.

Description

This command is a Perl script used to delete the configuration data of a RM pool.

If the *update_config* operand is **yes**, the RM server is notified of the change to access_control and the new value is used at the next client connection. If **no** is specified, a second action must be taken before the new value is used. If the RM server is not running, the action must be to start the RM server. If the RM server is running, the action would be to reconfigure the RM server.

After this script is executed, the pool data is deleted from the System Data Repository (SDR) and the /**etc/jmd_config.***syspar_name* file is updated to reflect the current configuration.

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment. Both operands are required.

The main use of this script is via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit delete_pool
```

Files

/**etc/jmd_config.***syspar_name*
Resource Manager configuration data file.

Related Information

Command: **jmcmi_createjmdconfig**

Examples

To delete Pool 1 and inform the RM server of these changes, enter:

```
jmcmi_deletepool 1 yes
```

jmcmi_servernodes

Purpose

jmcmi_servernodes – Defines the Resource Manager (RM) server list.

Syntax

jmcmi_servernodes [–**h**] *update_config {***yes** | **no***} 1st_server_node 2nd_server_node additional_server_nodes*

Flags

–**h** Displays usage information.

Operands

update_config
Specifies whether to notify the RM server when configuration changes are made. Valid options are **yes** or **no**.

1st_server_node
Specifies the first node in the RM server list. It correlates to the JM_LIST keyword found in the /**etc/jmd_config.***syspar_name* file.

2nd_server_node
Specifies the second node in the RM server list. It correlates to the JM_LIST keyword found in the /**etc/jmd_config.***syspar_name* file

additional_server_nodes
Specifies any remaining nodes that should be defined in the RM server list. It correlates to the JM_LIST keyword found in the /**etc/jmd_config.***syspar_name* file.

Description

This command is a Perl script used to define the SP nodes available for the RM to run its servers on. The order of the nodes listed is important. The RM always tries to start the server on the *1st_server_node* if that node is available and then continues trying nodes in the list in the order they are specified until an available node is found. For more information on the RM server list and servers, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*.

If the *update_config* operand is **yes**, the RM server is notified that a configuration change was made, and updates all of its configuration data. This data is used at the next client connection. If **no** is specified, a second action must be taken before the new configuration data is used. If the RM server is not running, the action must be to start the RM server. If the RM server is running, the action would be to reconfigure the RM server.

The *1st_server_node* operand must specify a single node name. The *2nd_server_node* operand supports two options: a single node name or the keyword **none**. If **none** is specified, no other node names can be defined following it. The *additional_server_nodes* operand supports the same two options except more than one node name can be specified. If no further nodes need to be defined, the keyword **none** must be specified.

This script must be executed by root from the control workstation and operates within the scope of the current system partition environment. All operands are required.

After this script is executed, the data is updated within the System Data Repository (SDR) and the /**etc/jmd_config.***syspar_name* file is updated with the new values for JM_LIST.

The main use of this script is via the System Management Interface Tool (SMIT). To use SMIT, enter:

```
smit server_list
```

Files

/**etc/jmd_config.***syspar_name*
Resource Manager configuration data file.

Related Information

Command: **jmcmi_createjmdconfig**

Examples

1. To define the RM server list to consist of four nodes and inform the RM server of these changes, enter:

   ```
   jmcmi_servernodes yes r07n01
   r07n10 r07n128 r07n150
   ```

2. To define the RM server list to consist of two nodes and not to inform the RM server of these changes, enter:

   ```
   jmcmi_servernodes no r07n01
   r07n10 none
   ```

   In this case, the RM needs to be informed later in order for this to take affect.

kadmin

Purpose

kadmin – Provides network access to authentication database administration functions.

Syntax

kadmin [–**u** *admin_name*] [–**r** *default_realm*] [–**m**]

Flags

–**u** Specifies a principal name to use instead of your AIX login name. This *admin_name* must be a valid AIX login name.

–**r** Specify if you want a realm other than the local realm to be the default.

–**m** Allows multiple requests without reauthentication (reentry of your administrative password).

Operands

None.

Description

This command provides an interactive interface to the primary authentication database. Administrators use **kadmin** to add new users and services to the database, and to change information about existing database entries. For example, an administrator can use **kadmin** to change a user's password. An administrator is a user with an **admin** instance whose name appears in at least one of the authentication administration Access Control Lists (ACLs).

The **kadmin** program communicates over the network with the **kadmind** program, which runs on the machine housing the primary authentication database. The **kadmind** program creates new entries and makes modifications to the database.

When you enter the **kadmin** command, the program displays a message that welcomes you and explains how to ask for help. Then **kadmin** waits for you to enter commands. After you enter a command, you are prompted to enter your admin password. If the –**m** option is used, you are prompted for your admin password only for the first command entered. You do not need to issue the **kinit** command prior to running this command because the necessary tickets are obtained automatically.

When using the **kadmin** command, the principal's expiration date and maximum ticket lifetime are set to the default values. To override the defaults, the root user must run the **kdb_edit** command to modify those attributes.

Use the **add_new_key** (or **ank** for short) command to add a new principal to the authentication database. The command requires the principal identifier as an argument. The identifier given can be fully qualified using the standard *name.instance@realm* convention. You are asked to enter your admin password and are then prompted twice to enter the principal's new password. If a realm is not specified, the local realm is used unless another was given on the command

line with the **r** flag. If no instance is specified, a null instance is used. If a realm other than the default realm is specified, you need to supply your admin password for the specified realm.

Use **change_password** to change a principal's password. The command requires the principal identifier as an argument. You are asked to enter your admin password and are then prompted twice to enter the principal's new password. The identifier given can be fully qualified using the standard *name.instance@realm* convention.

Use the **change_admin_password** to change your admin instance password. This command requires no arguments. It prompts you for your old admin password, then prompts you twice to enter the new admin password. If this is your first command, the default realm is used. Otherwise, the realm used in the last command is used.

Use **destroy_tickets** to destroy any admin tickets obtained by the **kadmin** command.

Use **list_requests** to get a list of possible commands.

Use **help** to display various **kadmin** help messages. If entered without an argument, **help** displays a general help message. You can get detailed information on specific **kadmin** commands by entering **help** *command_name*.

To quit the program, type **quit**.

Files

/var/kerberos/database/admin_acl.{add,get,mod}
Access Control List files.

Related Information

Commands: **add_principal**, **kadmind**, **kpasswd**, **ksrvutil**

Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Examples

The following contains an example of adding a user. To add a user, enter:

kadmin

```
Welcome to the Kerberos Administration Program, version 2
Type "help" if you need it.
admin:  help
Welcome to the Kerberos administration program.Type "?" to get
a list of requests that are available. You can get help on each of
the commands by typing "help command_name". Some functions of this
program requires an "admin" password from you. This is a password
private to you, that is used to authenticate requests from this
program. You can change this password with the "change_admin_password"
(or short form "cap") command. Good Luck!
admin:  ?

Available admin requests:

change_password, cpw        Change a user's password
change_admin_password, cap  Change your admin password
add_new_key, ank            Add new user to kerberos database
get_entry, get              Get entry from kerberos database
destroy_tickets, dest       Destroy admin tickets
help                        Request help with this program
list_requests, lr, ?        List available requests.
quit, exit, q               Exit program.

admin:  ank mroz
Admin password:
Password for mroz:
Verifying, please re-enter Password for mroz:
mroz added to database.
admin:  q
Cleaning up and exiting.
```

Note: Passwords are not echoed back to the user.

kadmind daemon

Purpose

kadmind – Contains the daemon for authentication database administration.

Syntax

kadmind [–**h**] [–**n**] [–**r** *realm*] [–**d** *db_name*] [–**f** *file_name*] [–**a** *acldir*]

Flags

–**h** Specifies that the **kadmind** command list the available subcommands and exit.

–**n** Specifies that the master key from the master key cache file be obtained. Otherwise, it prompts the user to enter the master key interactively.

–**r** Specifies that the **kadmind** command is to service a realm other than the local realm. *realm* is the authentication realm name.

–**d** Specifies an authentication database name other than the default. *db_name* is a directory path.

–**f** Specifies the log file in which the daemon records status and error messages.

–**a** Specifies a directory other than the default that contains the Access Control Lists. *acldir* is a directory path.

Note: Use of the –**r**, –**d**, and –**a** flags with values other than the system defaults is not supported on the SP system.

Operands

None.

Description

The **kadmind** daemon is the authentication database server for the password-changing and administration tools. It uses the master key for authorization.

The **kadmind** daemon listens for requests on the **kerberos_master**/**tcp** port. If this port is not defined in the /**etc**/**services** file, it uses port 751.

When performing requests on behalf of clients, **kadmind** checks access control lists (ACLs) to determine the authorization of the client to perform the requested action. Currently three distinct access types are supported:

- Addition (.add ACL file). A principal on this list can add new principals to the database.
- Retrieval (.get ACL file). A principal on this list can retrieve database entries.

  Note: A principal's private key is never returned by the get functions.
- Modification (.mod ACL file). A principal on this list can modify entries in the database.

Principals are always granted authorization to change their own password.

Files

/.k Master key cache file.

/var/kerberos/database/admin_acl.{add,get,mod}
Access Control List files.

/var/kerberos/database/principal.pag, /kerberos/database/principal.dir
Default files containing the authentication database.

/var/adm/SPlogs/kerberos/admin_server.syslog
Default log file.

Related Information

Commands: **add_principal**, **kadmin**, **kpasswd**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kdb_destroy

Purpose

kdb_destroy – Destroys the authentication database.

Syntax

kdb_destroy

Flags

None.

Operands

None.

Description

The **kdb_destroy** command removes the authentication database.

You first must reply **y** or **Y** to a prompt to confirm the request, or **kdb_destroy** exits without removing the database files.

This command can only be issued on the system on which the authentication database resides.

Note: This command does not remove database backup files created by the **kdb_util** command nor the **/.k** file created by the **kstash** command.

Files

/var/kerberos/database/principal.pag, /usr/kerberos/database/principal.dir
Files containing the authentication database.

Security

You must have root privilege to run this command.

Related Information

Command: **kdb_init**

kdb_edit

Purpose

kdb_edit – Edits the authentication database.

Syntax

kdb_edit [–**n**]

Flags

–**n** Specifies that the master key is obtained from the master key cache file. Otherwise, **kdb_edit** prompts the user to enter the master key interactively.

Operands

None.

Description

The **kdb_edit** command is used to create or change principals in the authentication database. It uses the master key for authorization.

After the master key is verified, **kdb_edit** begins a prompt loop. The user is prompted for the principal name and instance to be modified. If the entry is not found, the user can create it. After an entry is found or created, the user can set the password, expiration date, maximum ticket lifetime, and attributes. Default expiration dates, maximum ticket lifetimes, and attributes are presented in brackets. If the user presses return, the default is selected. There is no default password. The password RANDOM is interpreted specially, and if entered, the program selects a random key for the principal.

You should use random key generation only if you use the **kdb_edit** command to replace a deleted service principal (for example, **rcmd**.*host_name*).

If you enter a ticket lifetime value, it must be a number between 0 and 255. The actual maximum lifetime value that you choose will be between five minutes and 30 days. Refer to the *IBM Parallel System Support Programs for AIX: Administration Guide* for a complete list of the possible ticket lifetime values you can enter and the corresponding durations in days, hours, and minutes. The following list shows a representative sample with approximate durations:

```
Response to kdb_edit          Approximate Duration
       141                           1 day
       151                           2 days
       170                           1 week
       180                           2 weeks
       191                           1 month
```

After the entry has been created or changed, “Edit O.K.” is printed.

Files

/.k Master key cache file.

/var/kerberos/database/principal.pag, /usr/kerberos/database/principal.dir
Files containing the authentication database.

Security

You must have root privilege to run this command.

Related Information

Commands: **kadmin**, **kdb_init**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Examples

To add a service from host **mroz**, enter:

kdb_edit -n

```
Opening database...
Previous or default values are in [brackets],
enter return to leave the same, or new value.

Principal name: rcmd
Instance: mroz

<Not found>, Create [y] ? Y

Principal: rcmd, Instance: mroz, kdc_key_ver: 1
New Password:
Verifying, please re-enter
New Password:

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [1999-12-31] ?
Max ticket lifetime [255] ?
Attributes [0] ?
Edit O.K.
Program re-prompts for another principal "principal name:"

Principal name:

The program exits when no principal name is entered.
```

Note: Passwords are not echoed back to the user.

kdb_init

Purpose

kdb_init – Initializes the authentication database.

Syntax

kdb_init [*realm*]

Flags

None.

Operands

realm Specifies the realm name. If *realm* is not specified, the realm name is set to the local system's network domain name converted to uppercase characters.

Description

Use this command to initialize the authentication database, creating the necessary initial system principals.

After determining the realm to be created, the command prompts for a master key password. The user should choose a nontrivial, not easily-guessable password. The user **must** remember this password because it is used for other commands. The master key password is used to encrypt every encryption key stored in the database.

Files

/var/kerberos/database/principal.pag, /usr/kerberos/database/principal.dir
Files containing the authentication database.

Security

You must have root privilege to run this command.

Related Information

Commands: **kdb_destroy**, **kdb_edit**, **kdb_util**

Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kdb_util

Purpose

kdb_util – Contains the utility program for managing the authentication database.

Syntax

kdb_util *operation file_name*

Flags

None.

Operands

operation The *operation* must be one of the following:

load Initializes the database with the records described by the text contained in the file *file_name*. Any existing database is overwritten.

dump Dumps the database into a text representation in the file *file_name*.

slave_dump
Performs a database dump similar to the dump operation and creates a semaphore file to indicate to the propagation software that an update is available for distribution to secondary authentication servers.

new_master_key
Prompts for the old and new master key strings, and then dumps the database into a text representation in the file *file_name*. The keys in the text representation are encrypted in the new master key.

file_name Specifies the name of the file.

Description

The **kdb_util** command allows the user to perform various utility operations on the authentication database.

Files

/var/kerberos/database/principal.pag, /usr/kerberos/database/principal.dir
Files containing the authentication database.

<**data_file**>**.ok** Semaphore file created by the **slave_dump** operation.

Security

You must have root privilege to run this command.

Related Information

Commands: **kdb_init**, **kprop**, **kpropd**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kdestroy

Purpose

kdestroy – Destroys authentication tickets.

Syntax

kdestroy [–**f**] [–**q**]

Flags

–**f** Indicates that **kdestroy** should not display a status message.

–**q** Indicates that **kdestroy** should display a status message, but should not beep the terminal on an error.

Operands

None.

Description

The **kdestroy** command destroys the user's authentication tickets. The command writes zeros to the user's current ticket cache file and then removes the file from the file system. If the file does not exist or if an error occurs, a message is displayed. The current ticket file is determined by the KRBTKFILE environment variable. If the KRBTKFILE environment variable is undefined, the current ticket file is /**tmp**/**tkt***uid*, where *uid* specifies your user identification number. If **kdestroy** cannot destroy the ticket file, the command warns you by making your terminal beep. You can place the **kdestroy** command in your **.logout** file (C shell only) so that your tickets are destroyed automatically when you log out.

Files

/**tmp**/**tkt***uid* The default ticket file (*uid* is the decimal UID of the user).

Related Information

Commands: **kinit**, **klist**

Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kerberos daemon

Purpose

kerberos – Contains the authentication ticket-granting service daemon.

Syntax

kerberos [–**a** *max_age*] [–**l** *log_file*] [–**m**] [–**n**] [–**p** *pause_seconds*] [–**r** *realm*] [–**s**] [*database*]

Flags

–**a** Specifies the maximum database age. Its value must be between one hour and three days, in seconds. For slave servers, the default is one day. For the primary server, the default is not to check the age of the database.

–**l** Specifies the log file path name. (This is lowercase **l**, as in **l**ist.)

–**m** Prompts for the master key. If the –**m** option is not specified, the master key is obtained from the master key cache file.

–**n** Specifies that the age of the database against maximum not be checked. If desired, this option can override the default for secondary servers.

–**p** Specifies the pause interval. It must be between 5 and 3600 seconds. The default is to hang indefinitely on an error.

–**r** Allows the realm to be specified instead of assuming the local realm.

–**s** Indicates that this server is a secondary (backup) server.

Operands

database Contains the path name of the authentication database.

Note: Specification of a database path name other than the default, /**var**/**kerberos**/**database**/**principal**, is not supported on the SP system.

Description

kerberos is the daemon program that provides the Authentication Service and the Ticket Granting Service to client programs that want to obtain tickets for authenticated services.

The **kerberos** daemon listens for requests on the **kerberos4**/**upd** port. If this port is not defined in the /**etc**/**services** file, it uses port 750.

When you start the server (normally from **init**), you can specify a maximum age for the database files. This can be used to ensure that you do not start a secondary server with out-of-date information. This could occur in a situation where a secondary server system was down when a database update was scheduled.

kerberos daemon

Files

/var/kerberos/database/principal.pag, /var/kerberos/database/principal.dir
Files containing the authentication database.

/.k Master key cache file.

/var/adm/SPlogs/kerberos/kerberos.log,
/var/adm/SPlogs/kerberos/kerberos.slave_log
Log files.

Related Information

Commands: **kdb_init**, **kprop**, **kpropd**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kinit

Purpose

kinit – Contains the SP login authentication program.

Syntax

kinit [–**i**] [–**l**] [–**r**] [–**v**] [*name*]

Flags

–**i** Requests the command to prompt you for an instance, unless one is specified in the *name* operand.

–**l** Requests the command to prompt you for the ticket lifetime. If not specified, the ticket will have the maximum time allowed for the user. (This is lowercase **l**, as in **l**ist.)

–**r** Requests the command to prompt you for an authentication realm, unless one is specified in the *name* operand. This option lets you authenticate yourself within an authentication realm other than the local realm.

–**v** Specifies verbose mode. The name of the ticket file used is printed and a status message indicating the success or failure of your authentication attempt.

Operands

name Specifies your user principal identifier. The principal name can be qualified with an instance and/or realm *name.instance@realm*. Refer to the **Kerberos** command for details.

Description

The **kinit** command is used to authenticate the user's identify to the SP authentication service. All previous tickets are discarded.

When you use the **kinit** command without options, it prompts for your principal name and password, and tries to authenticate your identity within the local realm. If the specified principal name and password are correct, **kinit** retrieves your initial ticket and puts it in the ticket file specified by your KRBTKFILE environment variable. If the KRBTKFILE variable is undefined, your ticket is stored in the /**tmp**/**tkt***uid* file, where *uid* specifies your user identification number.

Note: These tickets are shared by all processes running under the user's IDs. The KRBTKFILE environment variable can be set to change the location of the ticket cache file.

If you specify the –**l** flag, the command prompts you to enter a ticket lifetime, in minutes. The actual value you enter will differ somewhat from the actual lifetime, because lifetimes are set to one of a discrete set of values ranging from five minutes to 30 days. **kinit** rounds the value you enter up to the next higher limit, and applies the maximum that is defined for your Kerberos principal. If you enter a value higher than your allowed limit, **kinit** does not indicate an error, but simply assigns your maximum lifetime in the ticket it creates. Refer to the *IBM Parallel System Support Programs for AIX: Administration Guide* for the complete list of

maximum lifetime values that the administrator can set. The following list shows a representative sample of lifetimes you can request:

```
Response to kinit prompt          Approximate duration
       1500                             1 day
       3000                             2 days
      10000                             1 week
      20000                             2 weeks
      43000                             1 month
```

Depending on your security policy, you may want to use the **kdestroy** command to destroy any active tickets before you end your login session. You can place the **kdestroy** command in your **.logout** file (C shell only) so that your tickets are destroyed automatically when you logout.

The KRBTKFILE environment variable is used to specify the ticket cache file used by **kinit** to store authentication tickets.

Files

/**tmp**/**tkt***uid* The default ticket file (*uid* is the decimal UID of the user).

Related Information

Commands: **kdestroy**, **klist**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

klist

Purpose

klist – Lists currently held authentication tickets.

Syntax

klist [–**s** | –**t**] [–**file** *name*] [–**srvtab**]

Parameters

–**s** Indicates silent mode. The **klist** command does not print the issue and expire times, the name of the tickets file, or the identity of the principal. This flag is ignored if **srvtab** is specified.

–**t** Indicates test mode. The **klist** command just checks for the existence of a nonexpired ticket-granting-ticket. If one is present, it exits with a status of 0. Otherwise, it exits with a status of 1. No output is displayed.

–**file** Specifies the name of a ticket cache file. When the –**file** option is not specified, the **klist** command uses the KRBTKFILE environment variable to determine the location of the ticket cache file. If KRBTKFILE is not set, /**tmp**/**tkt***uid* file is used, where *uid* is the AIX user ID. When **srvtab** is also specified, this flag specifies the name of the server key file whose contents are to be displayed.

–**srvtab** Specifies that the **klist** command is to list the contents of a server key file instead of a ticket cache file. If the **file** option is not specified, the default key file is /**etc**/**krb-srvtab**.

Operands

None.

Description

The **klist** command prints the principal name and the name of the file containing the user's tickets. It also lists the principal name, issue time, and expiration time for each service ticket held by the user. Principal names are listed in the form *name.instance@realm*. The period (.) is omitted if the instance is null and the at sign (@) is omitted if the realm is null.

Files

/**etc**/**krb.conf** Contains the name of the local realm.

/**etc**/**krb-srvtab** The default service key file.

/**tmp**/**tkt***uid* The default ticket file (*uid* is the decimal UID of the user).

Related Information

Commands: **kdestroy**, **kerberos**, **kinit**

Examples

1. This example shows a listing of the default ticket cache file for the root user (uid 0):

```
# klist
Ticket file:    /tmp/tkt0
Principal:  root.admin@XYZ.ABC.COM

  Issued           Expires          Principal
Nov 12 16:26:11  Dec 12 16:26:11  krbtgt.XYZ.ABC.COM@XYZ.ABC.COM
Nov 12 16:26:46  Dec 12 16:26:46  hardmon.cwksta@XYZ.ABC.COM
Nov 12 16:45:15  Dec 12 16:45:15  rcmd.cwksta@XYZ.ABC.COM
#
```

   The second line shows the Kerberos principal acting as client, to whom the tickets belong. This is the user principal you supplied to the **kinit** command, or the **rcmd**.*instance* service principal used by **rcmdtgt**. The list of tickets always begins with the ticket-granting-ticket. The others are service tickets; in this case for the System Monitor service on the control workstation (**hardmon**) and the SP Remote Command service also on the control workstation (**rcmd**).

2. This example shows the use of **klist** to display the key versions for service principals on an SP node:

```
# klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
-----------------------------------------------------
rcmd            node3fi         XYZ.ABC.COM       1
rcmd            node3tr         XYZ.ABC.COM       1
rcmd            node3sw         XYZ.ABC.COM       1
rcmd            node3en         XYZ.ABC.COM       1
#
```

   You can determine the versions of service keys in the authentication database by locating the entry for the target service principal in a dump of the SP authentication database. If you have secondary authentication servers, or if you use the procedure for backing up your database that is recommended in *IBM Parallel System Support Programs for AIX: Administration Guide*, the database dump can be found in file /**var**/**kerberos**/**database**/**slavesave** on the primary server host.

kpasswd

Purpose

kpasswd – Changes the principal's password.

Syntax

kpasswd [–**h**] [–**n** *user*] [–**i** *instance*] [–**r** *realm*] [–**u** *full_name*]

Flags

–**h** Specifies that **kpasswd** is to print a brief summary of the options and then exit.

–**n** Specifies the name to be used as the principal name rather than the user name of the user running **kpasswd**. (This is determined from the ticket file if it exists; otherwise, it is determined from the AIX login name.)

–**i** Specifies the instance to be used as the instance of the user principal, rather than a null instance.

–**r** Specifies the realm to be used as the realm rather than the local realm.

–**u** Specifies a fully qualified principal identifier in the form *name.instance@realm*.

Operands

None.

Description

The **kpasswd** command changes a principal's password.

It prompts for the principal's current password. If the old password is correct, the user is prompted twice for a new password. A message is printed indicating the success or failure of the password changing operation.

Related Information

Commands: **kadmin**, **kinit**, **passwd**

Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

kprop

Purpose

kprop – Specifies the network utility to propagate the authentication database to secondary servers.

Syntax

kprop [–**force**] [–**realm** *realm_name*] *data_file hosts_file*

Flags

–**force** Overrides the timestamp checking, forcing transmittal even if the database was not modified since last sent.

–**realm** Allows the realm to be specified instead of assuming the local realm.

Operands

data_file Specifies the file containing the dumped image of the authentication database produced by the **kdb_util slave_dump** command.

hosts_file Contains a list of secondary server hosts that provide backup to this server.

Description

The **kprop** command reads a list of secondary host names and connects to each one in turn using the **kprop** service provided by the **kpropd** program. The *data_file* (the authentication database) is transferred if it has been modified since it was last sent successfully.

Files

<**data_file**>**.ok** Semaphore file created by the **kdb_util slave_dump** operation.

Related Information

Commands: **kdb_util**, **kerberos**, **kpropd**

kpropd daemon

Purpose

kpropd – Contains the daemon to receive updates for a secondary authentication database.

Syntax

kpropd [–**r** *realm*] [–**s** *srvtab*] [–**l** *log_file*] [–**d** *database_name*] *file_name*

Flags

–**r** Overrides the default local realm.

–**s** Overrides the default srvtab name /**etc**/**krb-srvtab**.

–**l** Specifies a log file name to be used instead of the default. (This is lowercase **l**, as in **l**ist.)

–**d** Specifies the path name of the database.

Note: Use of the –**r**, –**s**, and –**d** flags with values other than the system defaults is not supported on the SP system.

Operands

file_name
Specifies the name of the file to receive from the transmitting host, then input to a **kdb_util load** command.

Description

kpropd runs as a daemon on secondary authentication database server hosts, listening for a TCP connection on the **krb_prop** service.

The **kpropd** daemon listens for requests on the **krb_prop**/**tcp** port. If this port is not defined in the /**etc**/**services** file, it uses port 754. It validates the connection, which must be from an administrative host as defined in the **krb.conf** file for the local realm. The service name used for mutual authentication is **rcmd**.

Files

/**etc**/**krb.conf** Contains the name of the local realm.

/**etc**/**krb-srvtab** Default server key file.

/**var**/**kerberos**/**database**/**principal.pag**, /**var**/**kerberos**/**database**/**principal.dir**
Default location of database files.

/**var**/**adm**/**SPlogs**/**kerberos**/**kpropd.log**
Log file.

kpropd daemon

Related Information

Commands: **kdb_util**, **kerberos**, **kprop**

kshd daemon

Purpose

kshd – Provides the server function for remote command execution.

Syntax

kshd kshd [–**s**] *program_name*

The **kshd** daemon is normally started by the **inetd** daemon.

Flags

–**s** Turns on socket-level debugging.

Operands

program_name
Specifies the program name.

Description

The **kshd** daemon is the server for the SP **rcp** and **rsh** commands. It provides remote execution of shell commands. These commands are based on requests from privileged sockets on trusted hosts. The shell commands must have user authentication. The **kshd** daemon listens at the socket defined in the /**etc**/**services** file and in the InetServ object class with the command entry.

The **kshd** daemon is started by default. The **inetd** daemon no longer reads the /**etc**/**inetd.conf** file, although this file still exists. Instead, the **inetd** daemon gets its information from the InetServ object class (stored in the AIX Object Data Manager). This object class is a combination of the information in the /**etc**/**inetd.conf** file and the /**etc**/**services** file. InetServ is created at install time from information in these two files.

If you have already set up the **kshd** daemon using the /**etc**/**inetd.conf** file, or if you are accustomed to using this file and want to continue doing so, you can. However, the InetServ object class and the /**etc**/**services** and /**etc**/**inetd.conf** files must be kept synchronized. If you modify the /**etc**/**inetd.conf** or the /**etc**/**services** file, you need to run the **inetimp** command to apply those changes to the InetServ object class. Then run the **refresh** –**s inetd** command to immediately update the **inetd** daemon.

When the **kshd** daemon receives a service request, it initiates the following protocol:

1. The **kshd** daemon checks the source port number for the request. If the port number is not in the range 0 through 1023, the **kshd** daemon terminates the connection.
2. The **kshd** daemon reads characters from the socket up to a null byte. The string read is interpreted as an ASCII number (base 10). If this number is nonzero, the **kshd** daemon interprets it as the port number of a secondary stream to be used as standard error. A second connection is created to the

specified port on the client host. The source port on the local host is also in the range of 0 through 1023.

3. The **kshd** daemon uses the source address of the initial connection request to determine the name of the client host. If the name cannot be determined, the **kshd** daemon uses the dotted decimal representation of the client host's address.
4. The **kshd** daemon retrieves the following information from the initial socket:
   a. A ticket/authenticator pair is retrieved on the initial socket.
   b. A null-terminated string of at most 16 bytes is interpreted as the user name of the user on the client host.
   c. A null-terminated string of at most 16 bytes is interpreted as the user name to be used on the local server host.
   d. Another null-terminated string is interpreted as a command line to be passed to a shell on the local server host.
5. The **kshd** daemon attempts to validate the user using the following steps:
   a. The **kshd** daemon looks up the local user name in the **/etc/passwd** file and tries to switch to the home directory (using the **chdir** system call). If either the lookup or the directory change fails, the **kshd** daemon terminates the connection.
   b. The **.klogin** file in the home directory is used to determine access to the account (via **kuserok**) by the principal named in the ticket/authenticator. If this authorization check fails, the connection is terminated.
6. After **kshd** validates the user, the **kshd** daemon returns a null byte on the initial connection and passes the command line to the user's local login shell. The shell then inherits the network connections established by the **kshd** daemon.

   The **kshd** daemon first attempts to authenticate the requester using the key of the **rcmd** service instance whose name is the local host name. If that fails, **kshd** attempts to authenticate using each of the service keys for the other instances of the service. A separate service instance exists for each network interface through which the server may be reached.

This daemon does not support encryption by users.

Files

/usr/lpp/ssp/rcmd/etc/kshd
Executable file.

/etc/services Defines Internet socket assignments.

$HOME/.klogin
Defines equivalent remote users.

Prerequisite Information

- An understanding of the TCP/IP daemons
- Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for an overview.
- Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Related Information

SP Commands: **kinit**, **rcp**, **rsh**

AIX Commands: **inetimp**, **rsh**

AIX Daemon: **inetd**

ksrvtgt

Purpose

ksrvtgt – Obtains a ticket-granting-ticket using a service key.

Syntax

ksrvtgt *name instance* [[*realm*] *srvtab*]

Flags

None.

Operands

name
instance
realm Specifies the principal as *name.instance@realm* (where realm defaults to the local realm defined in /**etc**/**krb.conf**).

srvtab Specifies the service key file to use (defaults to /**etc**/**krb-srvtab**).

Note: Specification of a **srvtab** file other than the system default is not supported on the SP system.

Description

The **ksrvtgt** command retrieves a ticket-granting-ticket with a lifetime of five minutes, decrypts the response using the service key found in the service key file, and stores the ticket in the standard ticket cache.

This command is intended primarily for use in shell scripts and other batch-type facilities.

The KRBTKFILE environment variable is used to specify the ticket cache file used by **ksrvtgt** to store authentication tickets.

Files

/**etc**/**krb.conf** Contains the name of the local realm.

/**etc**/**krb-srvtab** The default service key file.

/**tmp**/**tkt**<**uid**> The default ticket file.

Security

You must have root privilege to run this command.

Related Information

Commands: **kdestroy**, **kinit**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

ksrvutil

Purpose

ksrvutil – Manipulates a server key file.

Syntax

ksrvutil [–**afs** | –**krb**] [–**k**] [–**i**] [–**f** *file_name*] *operation*

Flags

–**afs** Indicates that the authentication database is being managed by AFS, and that the key file should be constructed to work with AFS.

–**krb** Indicates that the authentication database is being managed by a server compatible with the MIT version of Kerberos, and that the key file should be constructed to work with that version.

If neither –**afs** nor –**krb** are specified, the value of the System Data Repository (SDR) **authent_server** attribute is used. If the value of the SDR **authent_server** attribute cannot be obtained, the default is –**krb**.

–**k** When specified for the list operation, keys are also displayed. For the change operation, the old and new keys are displayed. For the add operation, the key is displayed.

–**i** Prompts for yes or no before changing each key.

–**f** For all operations, specifies the server key file to update. The default is **/etc/krb-srvtab**.

Note: Specification of a **srvtab** file other than the system default is not supported on the SP system.

Operands

operation The operation must be one of the following:

list Lists the version number and principal name in the server key file.

change Changes all the keys in the server key file.

add Adds a server principal name and key to the server key file. The command prompts for name, instance, realm, and key version number, and asks for a password. The **ksrvutil** command then converts the password to a key and appends the key file with the new information.

delete Deletes keys in the key file. The user is prompted before deleting each key.

Description

The **ksrvutil** command allows an administrator to list or change keys currently in the key file or to add new keys to the keyfile.

The **ksrvutil** command always backs up the key file before making any changes. If **ksrvutil** fails during a change or add operation, you can recover a usable key file by appending the workfile containing the new and changed keys, *file_name*.work to

the backup copy or the original, *file_name*.old, and replacing the key file *file_name* with the result, for example:

```
cat /etc/krb-srvtab.old /etc/krb-srvtab.work >/etc/krb-srvtab
```

The recovered key file can be used, but it may contain some out-of-date keys.

Files

/**etc**/**krb-srvtab** Default server key file.

Security

You must have root privilege to run this command.

Related Information

Commands: **kadmin**, **ksrvtgt**, **rcmdtgt**

kstash

Purpose

kstash – Saves the system's authentication master key.

Syntax

kstash

Flags

None.

Operands

None.

Description

The **kstash** command saves the system's authentication database master key in the master key cache file. The user is prompted to enter the master key (the same one as specified to **kdb_init**) to verify the authenticity of the key and authorize caching it.

Files

/.k Master key cache file.

/var/kerberos/database/principal.pag, /var/kerberos/database/principal.dir
Files containing the authentication database.

Security

You must have root privilege to run this command.

Related Information

Command: **kdb_init**

locate_jm

Purpose

locate_jm – Identifies the nodes on which the primary and backup Resource Manager servers are running within the current system partition.

Syntax

locate_jm

Flags

None.

Operands

None.

Description

Use this command to identify the nodes on which the primary and backup Resource Manager servers are running. This command must be executed by root and operates within the scope of the current system partition environment.

Files

/usr/lpp/ssp/bin/locate_jm
Path name of this command.

Related Information

Commands: **jm_config**, **jm_install_verify**, **jm_start**, **jm_status**, **jm_stop**

Examples

To locate the Resource Manager servers within the current system partition, enter:

```
locate_jm
```

You should receive a response similar to the following:

```
locate_jm: The primary JM server is running on r07n01.
locate_jm: The backup JM server is running on r07n03.
```

lppdiff

Purpose

lppdiff – Queries installed Licensed Program Products (LPPs) on a group of hosts.

Syntax

lppdiff [–**Gva**] [–**l** *login*] [–**w** *collective*] [–**f** *fanout*]
[*fileset* [*fileset* ...] | **all**]

lppdiff [–**h**]

Flags

–**G** Expands the scope of the –**a** flag to include all nodes in the SP system. The –**G** flag is meaningful only if used in conjunction with the –**a** flag.

–**v** Verifies hosts before adding to the working collective. If this flag is set, each host to be added to the working collective is checked before being added.

–**a** Specifies that the System Data Repository (SDR) **initial_hostname** field for all nodes in the current system partition be added to the working collective. If –**G** is also specified, all nodes in the SP system are included.

–**l** Specifies a remote user name under which to execute the query. If –**l** is not used, the remote user name is the same as your local user name.

–**w** Specifies a list of host names, separated by commas, to include in the working collective. Both this flag and the –**a** flag can be included on the same command line. Duplicate host names are included only once in the working collective.

–**f** Specifies a fanout value. The default value is 64. This indicates the maximum number on concurrent rsh's to execute. Sequential execution can be specified by indicating a fanout value of 1. The fanout value is taken from the FANOUT environment variable if the –**f** flag is not specified, otherwise the default is used.

–**h** Displays usage information.

Operands

fileset Specifies the LPP to query. Using **all** for this operand will query all LPPs installed on the host.

Description

Use this command to query the status of installed LPPs on a group of hosts. The output from each host is collected and identical results are compressed to a single result, listing all the host names that had that result.

The **dsh** command is used to execute the queries on the remote hosts. The **lslpp** command is used to get the status of the installed LPPs on the remote hosts. The **lslpp** command is called on each host with the –**l**, –**a**, –**c**, and –**q** flags.

Output from the **lppdiff** command consists of one entry for each unique LPP listing information about that LPP. Each LPP's entry is followed by a list of all hosts that have that LPP installed. An LPP is considered unique if any one of the components in its description differ from that of another. For example, consider two hosts that both have ssp.basic installed. On host 1, it is in the APPLY state and on host 2, it is in the COMMITTED state. These LPPs are considered unique and, therefore, each will get its own set of output from **lppdiff**.

The flags for **lppdiff** are used to direct the **dsh** command to certain hosts and to control its behavior. See the **dsh** command for details on these flags and how to use them.

The *fileset* operand to **lppdiff** can be one of two things. It can either be **all** which queries and displays information about all LPPs installed on the specified hosts, or it can be the name of a file set to query on the specified hosts. The "*" character can be used to specify multiple file sets. For example, **lppdiff –Ga ssp.*** queries any file sets starting with "ssp." on all hosts in the system.

Examples

1. To query LPP information for ssp.basic on all nodes in the current system partition, enter:

   ```
   [k22s] > lppdiff -a ssp.basic
   ```

   You should receive output similar to the following:

   ```
   ---------------------------------------------------------------------------
         Name        Path               Level      PTF     State     Type
   ---------------------------------------------------------------------------
   LPP:  ssp.basic   /etc/objrepos      2.2.0.0            COMMITTED   I
   From: k22n09.ppd.pok.ibm.com k22n10.ppd.pok.ibm.com k22n11.ppd.pok.
            ibm.com
         k22n12.ppd.pok.ibm.com k22n13.ppd.pok.ibm.com
   ---------------------------------------------------------------------------
   LPP:  ssp.basic   /usr/lib/objrepos  2.2.0.0            COMMITTED   I
   From: k22n09.ppd.pok.ibm.com k22n10.ppd.pok.ibm.com k22n11.ppd.pok.
            ibm.com
         k22n12.ppd.pok.ibm.com k22n13.ppd.pok.ibm.com
   ```

2. To query LPP information for all options starting with X11.base on a specific node, enter:

```
[k22s] > lppdiff -w k22n01 X11.base*
-------------------------------------------------------------------------
      Name          Path               Level      PTF    State     Type
-------------------------------------------------------------------------
LPP:  X11.base.rte  /etc/objrepos      4.1.4.0           COMMITTED   I
From: k22n01
-------------------------------------------------------------------------
LPP:  X11.base.smt  /etc/objrepos      4.1.4.0           COMMITTED   I
From: k22n01
-------------------------------------------------------------------------
LPP:  X11.base.comm /usr/lib/objrepos  4.1.0.0           COMMITTED   I
      on
From: k22n01
-------------------------------------------------------------------------
LPP:  X11.base.lib  /usr/lib/objrepos  4.1.4.0           COMMITTED   I
From: k22n01
-------------------------------------------------------------------------
LPP:  X11.base.rte  /usr/lib/objrepos  4.1.4.0           COMMITTED   I
From: k22n01
-------------------------------------------------------------------------
LPP:  X11.base.smt  /usr/lib/objrepos  4.1.4.0           COMMITTED   I
From: k22n01
```

3. To query LPP information for ssp.clients and ssp.bogus (a nonexistent file set) on all nodes in the system, enter:

lppdiff

```
[k22s] > lppdiff -Ga ssp.clients ssp.bogus
-------------------------------------------------------------------------
      Name          Path               Level      PTF    State     Type
-------------------------------------------------------------------------
LPP:  ssp.clients   /etc/objrepos      2.1.0.0           COMMITTED   I
From: k22n03.ppd.pok.ibm.com k22n04.ppd.pok.ibm.com k22n07.ppd.pok.
         ibm.com
      k22n08.ppd.pok.ibm.com
-------------------------------------------------------------------------
LPP:  ssp.clients   /etc/objrepos      2.1.0.6           APPLIED     F
From: k22n03.ppd.pok.ibm.com k22n04.ppd.pok.ibm.com k22n07.ppd.pok.
         ibm.com
      k22n08.ppd.pok.ibm.com
-------------------------------------------------------------------------
LPP:  ssp.clients   /etc/objrepos      2.2.0.0           COMMITTED   I
From: k22n01.ppd.pok.ibm.com k22n05.ppd.pok.ibm.com k22n06.ppd.pok.
         ibm.com
      k22n09.ppd.pok.ibm.com k22n10.ppd.pok.ibm.com k22n11.ppd.pok.
         ibm.com
      k22n12.ppd.pok.ibm.com k22n13.ppd.pok.ibm.com
-------------------------------------------------------------------------
LPP:  ssp.clients   /usr/lib/objrepos  2.1.0.0           COMMITTED   I
From: k22n03.ppd.pok.ibm.com k22n04.ppd.pok.ibm.com k22n07.ppd.pok.
         ibm.com
      k22n08.ppd.pok.ibm.com
-------------------------------------------------------------------------
LPP:  ssp.clients   /usr/lib/objrepos  2.1.0.6           APPLIED     F
From: k22n03.ppd.pok.ibm.com k22n04.ppd.pok.ibm.com k22n07.ppd.pok.
         ibm.com
      k22n08.ppd.pok.ibm.com
-------------------------------------------------------------------------
LPP:  ssp.clients   /usr/lib/objrepos  2.2.0.0           COMMITTED   I
From: k22n01.ppd.pok.ibm.com k22n05.ppd.pok.ibm.com k22n06.ppd.pok.
         ibm.com
      k22n09.ppd.pok.ibm.com k22n10.ppd.pok.ibm.com k22n11.ppd.pok.
         ibm.com
      k22n12.ppd.pok.ibm.com k22n13.ppd.pok.ibm.com

======================== Errors ===================================
Error: /bin/lslpp: Fileset ssp.bogus not installed.
From:  k22n01.ppd.pok.ibm.com k22n05.ppd.pok.ibm.com k22n06.ppd.pok.
          ibm.com
       k22n03.ppd.pok.ibm.com k22n04.ppd.pok.ibm.com k22n07.ppd.pok.
          ibm.com
       k22n08.ppd.pok.ibm.com k22n09.ppd.pok.ibm.com k22n10.ppd.pok.
          ibm.com
       k22n11.ppd.pok.ibm.com k22n12.ppd.pok.ibm.com k22n13.ppd.pok.
          ibm.com
-------------------------------------------------------------------------
```

lshacws

Purpose

lshacws – Gets the state of the control workstation.

Syntax

lshacws

Flags

None.

Operands

None.

Description

Use this command to print the current state of the control workstation. It prints to standard output a number string that indicates the state of the primary or backup control workstation and whether the control workstation is a high availability configuration.

This command is valid only when issued on the control workstation.

When the command is executed and the calling process is not on a control workstation, an error occurs.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

Exit Values

0 Indicates successful completion of the command.

1 Indicates that the command could not obtain the control workstation state.

2 Indicates that the command retrieved a control workstation state that was not valid.

3 Indicates that the command was not executed on a control workstation.

The following are the valid printed values and their defined control workstation state:

0 Indicates that the configuration is not an HACWS configuration, but is a control workstation.

1 Indicates that this is the primary control workstation, but not the active control workstation.

2 Indicates that this is the primary and active control workstation.

16 Indicates that this is the backup control workstation and not the active control workstation.

32 Indicates that this is the backup and active control workstation.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on the HACWS option.

Location

/usr/bin/lshacws

Related Information

Command: **sethacws**

Subroutines: **hacws_set**, **hacws_stat**

Examples

1. To determine if a node is a backup and active control workstation, enter:

   ```
   lshacws
   Results: 32
   ```

2. To determine if a node is a backup and inactive control workstation, enter:

   ```
   lshacws
   Results: 16
   ```

3. To determine if a node is a primary and active control workstation, enter:

   ```
   lshacws
   Results: 2
   ```

4. To determine if a node is a primary and inactive control workstation, enter:

   ```
   lshacws
   Results: 1
   ```

5. To determine if a node is a control workstation but not an HACWS configuration, enter:

   ```
   lshacws
   Results: 0
   ```

6. To determine if a node is not a control workstation, enter:

   ```
   lshacws
   Results: An error occurs and the exit value = 3
   ```

lshsd

Purpose

lshsd – Displays configured data striping device (HSD) for the IBM Virtual Shared Disks and their characteristics.

Syntax

lshsd [–**l** | –**s**] [*hsd_name* ...]

Flags

–**l** Lists the minor number, the stripe size, the number of IBM Virtual Shared Disks, the name of the data striping device, and the underlying IBM Virtual Shared Disks. (This is lowercase **l**, as in **l**ist.)

–**s** Displays the statistics of reads and writes on underlying IBM Virtual Shared Disks in HSDs.

Operands

hsd_name **...** Specifies an HSD for the IBM Virtual Shared Disk.

Description

This command displays the configured HSDs. If a list of HSDs follow the flag then information about them is displayed. **lshsd** without any arguments or flag lists the names of all the HSDs currently configured.

Files

/usr/lpp/csd/bin/lshsd Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsd**, **ucfghsd**

Examples

1. To list all the configured HSDs, enter:

   ```
   lshsd
   ```

   The system displays a message similar to the following:

   ```
   hsd1
   hsd2
   hsd3
   ⋮
   ```

2. To list HSDs and their characteristics, enter:

   ```
   lshsd -l hsd1 hsd2
   ```

   The system displays a message similar to the following:

```
HSD_name=hsd1 Stripe_size=32768 Hsd_minorno=1 numVsds=2
vsd.rlv01
vsd.rlv02

HSD_name=hsd2 Stripe_size=32768 Hsd_minorno=1 numVsds=3
vsd.rlv03
vsd.rlv04
vsd.rlv05
```

3. To list statistical information about data striping device **hsd1**, enter:

   ```
   lshsd -s hsd1
   ```

   The system displays a message similar to the following:

   ```
               9     hsd parallelism
               0     READ requests not at page boundary
               0     WRITE requests not at page boundary
   HSD_name=hsd1 Stripe_size=4096 HSD_minorno=1 numVSDs=2
   option=protect_lvcb
   number_read number_write vsd_name
       16          16       vsdn01v1
       16          16       vsdn02v1
   ```

lsvsd

Purpose

lsvsd – Displays configured IBM Virtual Shared Disks and their characteristics.

Syntax

lsvsd [–**l** | –**s**] [–**i**] [*vsd_name*...]

Flags

–**l** Lists the name of the IBM Virtual Shared Disk, the minor number, the state, the current server node number, and, at the server only, the major and minor number of the logical volume. (This is lowercase **l**, as in **l**ist.)

The state field can have one of the following values:

STP Stopped
SUS Suspended
ACT Active

This flag is not compatible with the –**s** flag.

–**s** Lists usage statistics about the IBM Virtual Shared Disks. It lists the number of local logical read and write operations, the number of remote logical read and write operations, the number of client logical read and write operations, the number of physical reads and writes, the number of cache hits for read, and the number of 512-byte blocks read and written. The number of blocks read and written is cumulative, so issue **ctlvsd -V** to reset this count before measuring it.

The local logical operations are requests which were made by a process executing at the local node, whereas the remote logical operations were made by a process executing on a remote node. *Client operations* are those local logical requests that cannot be satisfied locally, and have to be sent to a remote node. *Physical* operations are those server operations which must be passed to the underlying disk device. Cache read hits are those server reads which do not require a device read, because the read operation was satisfied from the IBM Virtual Shared Disk cache.

This flag is not compatible with the –**l** flag.

–**i** Lists the “node to IP address” map that is currently utilized by the IBM Virtual Shared Disk driver.

Operands

vsd_name Specifies an IBM Virtual Shared Disk.

Description

The **lsvsd** command displays information about IBM Virtual Shared Disks currently configured on the node on which the command is run. If a list of IBM Virtual Shared Disks follows the flags, information about those IBM Virtual Shared Disks is displayed. **lsvsd** with no arguments or flags lists the names of all the IBM Virtual Shared Disks currently configured on the node.

The **lsvsd** command displays information about both the configuration and the usage of an IBM Virtual Shared Disk.

You can use the System Management Interface Tool (SMIT) to run the **lsvsd** command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Show All Managed IBM Virtual Shared Disk Characteristics option.

Files

/usr/lpp/csd/bin/lsvsd Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Examples

1. To list all IBM Virtual Shared Disks in the system, enter:

   ```
   lsvsd
   ```

   The system displays a message similar to the following:

   ```
   vsd00
   vsd01
   ⋮
   ```

2. To list IBM Virtual Shared Disks and their characteristics, enter:

   ```
   lsvsd -l
   ```

   The system displays a message similar to the following:

   ```
   minor    state  server  lv_major lv_minor   vsd_name          option
   83       STP     -1        0       0        vsdn08v3          cache
   84       STP     -1        0       0        vsdn08v4          nocache
   ```

3. To list statistics about IBM Virtual Shared Disks and precede the column output with a header, enter:

   ```
   lsvsd -s
   ```

   The system displays a message similar to the following:

   ```
   lc-rd lc-wt rm-rd rm-wt c-rd c-wt  p-rd  p-wt  h-rd   br   bw  vsd_name
      84    84  2858   169    0    0   348   253  2605  164  184  vsd.vsd1
       0     0     0     0    0    0     0     0     0    0    0  vsd.rl01
       0     0     0     0    0    0     0     0     0    0    0  vsd.rl02
   ```

The following table spells out the names of the headers used in the displays for the **-l** and **-s** options:

Header	Meaning
minor	IBM Virtual Shared Disk minor number
state	State of this IBM Virtual Shared Disk:*active, stopped, suspended*
server	Primary node for this IBM Virtual Shared Disk
lv major	Logical volume major number
lv minor	Logical volume minor number
vsd_name	Name of this IBM Virtual Shared Disk
option	Option:*cache* or *nocache*
lc-rd	Local logical reads
lc-wt	Local logical writes
rm-rd	Remote logical reads
rm-wt	Remote logical writes
c-rd	Client logical reads
c-wt	Client logical writes
p-rd	Physical reads
p-wt	Physical writes
h-rd	Reads from cache
br	Blocks read
bw	Blocks written

mkamdent

Purpose

mkamdent – Creates entries in the Amd source file (/**etc**/**amd**/**amd-maps**/**amd.u**) used in the creation of the Amd maps.

Syntax

mkamdent [–**s** *server_path*] *user_names*

Flags

–**s** *server_path*
: Specifies the location from which the users' home directory is served. The format is *server_name: base_path*.

Operands

user_names
: A list of users to add to the source file, separated by spaces.

Examples

To create Amd entries in the **amd.u** map for multiple users:

```
mkamdent -s hostx:/home/hostx john ken pat paul ron
```

This assumes the following directories already exist on **hostx**:

- /**home**/**hostx**/**john**
- /**home**/**hostx**/**ken**
- /**home**/**hostx**/**pat**
- /**home**/**hostx**/**paul**
- /**home**/**hostx**/**ron**

mkconfig

Purpose

mkconfig – Creates the **config_info** file for each of the boot/install server's clients on the server.

Syntax

mkconfig

Flags

None.

Operands

None.

Description

Use this command to make the **config_info** files for all the clients of a boot/install server if the client is not set to boot from disk. The **mkconfig** command is intended to run only on the server node. This command creates a **config_info** file named /**tftpboot**/*host_name***.config_info** file.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–1 Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mkconfig

Related Information

Commands: **setup_server**

Examples

To make the **config.info** files for all boot/install clients of a server, enter on the server:

```
mkconfig
```

mkinstall

Purpose

mkinstall – Creates the **install_info** file for each of the server's clients on the server.

Syntax

mkinstall

Flags

None.

Operands

None.

Description

Use this command on the server node to make the **install_info** files for all clients of a boot/install server. The **mkinstall** command creates a **/tftpboot/***host_name***.install_info** file.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–1 Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mkinstall

Related Information

Commands: **setup_server**

Examples

To make the **install.info** files for all boot/install clients of a server, enter on the server:

```
mkinstall
```

mknimclient

Purpose

mknimclient – Makes a node a Network Installation Management (NIM) client of its boot/install server.

Syntax

mknimclient –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to be configured as clients of their boot/install servers. The *node_list* is a comma-separated list of node numbers.

Operands

None.

Description

Use this command to define a node as a NIM client. This is accomplished by determining the node's boot/install server from the System Data Repository (SDR) and configuring that client node as a NIM client on that server. When complete, the NIM configuration database on the server contains an entry for the specified client.

Notes:

1. This command results in no processing on the client node.
2. The assignment of a boot/install server for a node must first be made using **spbootins**.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mknimclient

Related Information

Commands: **delnimclient**, **setup_server**

Examples

To define nodes 1, 3, and 5 as NIM clients of their respective boot/install servers, enter:

```
mknimclient -l 1,3,5
```

mknimint

Purpose

mknimint – Creates the necessary Network Installation Management (NIM) interfaces on a NIM master.

Syntax

mknimint –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes on which to perform this operation. The *node_list* is a comma-separated list of node numbers. These nodes should have been previously configured as NIM masters (see the **mknimmast** command).

Operands

None.

Description

Use this command to define to NIM new Ethernet network adapters and interfaces on the control workstation and boot/install servers. On the control workstation, any networks not previously defined are defined and NIM interfaces added. On a boot/install server, all the Ethernet networks and interfaces are defined; it then defines all token ring and Ethernet networks that are known on the control workstation (with the **netstat** –**ni** command) and defines interfaces for them as well. This is so that resources like the **lppsource** can be served from the control workstation to a client node by the boot/install server if the client and control workstation are on the same subnetwork.

To serve a resource to a client that is not on the same subnetwork as the control workstation, routing is required. Routing is done in **mknimclient**.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mknimint

Related Information

Commands: **setup_server**

Examples

To make NIM interface definitions for nodes 1, 3, and 5, enter:

```
mknimint -l 1,3,5
```

mknimmast

Purpose

mknimmast – Configures a node as a Network Installation Management (NIM) master.

Syntax

mknimmast –**h** –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to be configured as NIM masters. The *node_list* is a comma-separated list of node numbers.

Operands

None.

Description

Use this command to define a boot/install server node as a NIM master for the subsequent installation of client nodes. It verifies that the listed nodes are defined as boot/install servers in the System Data Repository (SDR). It then installs the NIM master AIX file sets and configures the nodes as NIM masters.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mknimmast

Related Information

Commands: **delnimmast**, **setup_server**

Examples

To define nodes 1, 3, and 5 as NIM masters, enter:

```
mknimmast -l 1,3,5
```

mknimres

Purpose

mknimres – Creates the necessary Network Installation Management (NIM) resources on a NIM master.

Syntax

mknimres –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes on which to perform this operation. The *node_list* is a comma-separated list of node numbers. These nodes should have been previously configured as NIM masters (see **mknimmast**).

Operands

None.

Description

Use this command to make all the NIM resources for installation, diagnostics, migration, and customization. No resources are allocated to client nodes. The set of resources needed is determined from the list of client nodes found in the System Data Repository (SDR) for the *node_list*. Any required AIX install and **mksysb** images are defined as NIM resources. For boot/install server nodes, NIM Shared Product Object Tree (SPOT) directories are created and **mksysb** images are copied, as required. Because of the large data volumes required for SPOTs and install images, all checking is done before copying data.

Creation of the NIM lppsource resource on a boot/install server will result in **setup_server** creating a lock in the lppsource directory on the control workstation.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/mknimres

Related Information

Commands: **setup_server**

Examples

To make NIM resources for boot/install servers 1, 3, and 5, enter:

```
mknimres -l 1,3,5
```

ngaddto

Purpose

ngaddto – Adds nodes and node groups to the definition list of the destination node group.

Syntax

ngaddto [–**h**] | [–**G**] *dest_nodegroup*
nodenum | *nodegroup* [*nodenum* | *nodegroup*] ...

Flags

–**h** Displays usage information.

–**G** Specifies that the destination node group is global.

Operands

dest_nodegroup
Specifies the node group to receive the new additions.

nodenum Specifies a node to add to the definition list of the destination node group. This is supplied as a space-delimited list of node numbers.

nodegroup Specifies a named node group to add to the definition list of the destination node group. Node groups are given as a space-delimited list. Node numbers and node group names being added to the destination node group can be intermixed.

Description

Use this command to add nodes and node groups to the definition list of the destination node group. If the –**G** flag is specified, the destination node group must be global. If the –**G** flag is not specified, the destination node group must belong to the current system partition. If the destination node group does not exist, you will receive an error. If the destination node group or *nodegroup* is a name that is not valid, you will receive an error. Nodes and node groups that do not currently exist can be added to the destination node group. When the node group is resolved by the **ngresolve** command, nonexistent members are ignored.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to modify named node groups.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the "Managing Node Groups" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngaddto

Related Information

Commands: **ngcreate**, **ngdelete**, **ngdelfrom**, **ngfind**, **nglist**, **ngnew**, **ngresolve**

Examples

1. To add nodes 1 and 3 and node group ngb to the definition list of node group nga, enter:

   ```
   ngaddto nga 1 3 ngb
   ```

2. To add nodes 1 and 16 and global node group g2 to the global definition list of node group g1, enter:

   ```
   ngaddto -G g1 1 16 g2
   ```

ngclean

Purpose

ngclean – Cleans up a node group, removing references to nodes and node groups that are not in the current system partition. Node groups with empty definition lists will be deleted.

Syntax

ngclean [–**h**] | [–**G**] [–**r**] {–**a** | *nodegroup* [*nodegroup* ...]}

Flags

–**h** Displays usage information.

–**a** Cleans up all node groups in the current system partition or all system-wide node groups if the –**G** flag is also specified.

–**r** Does not modify node groups. Issues a report on how node groups would be affected by running this command (without the –**r** option).

–**G** Examines global node groups.

Operands

nodegroup Specifies the node groups to be cleaned. If the –**a** flag is provided, all node groups will be cleaned and no node groups should be specified.

Description

Use this command to examine node group definition lists and to remove references to nodes and node groups that do not exist in the current system partition or the SP system if –**G** is supplied. Node groups with empty definition lists will be deleted. If the –**r** flag is specified, the nodes and node groups will not be removed, but a report will be generated.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to modify a named node group.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the “Managing Node Groups” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngclean

Related Information

Command: **ngaddto**, **ngcreate**, **ngdelete**, **ngdelfrom**, **ngfind**, **nglist**, **ngnew**, **ngresolve**

Examples

1. To clean up all system node groups, enter:

   ```
   ngclean -Ga
   ```

2. To clean up the node group my.ng in the current system partition, enter:

   ```
   ngclean my.ng
   ```

ngcreate

Purpose

ngcreate – Creates and optionally populates a named node group.

Syntax

ngcreate [–**h**] | [–**s** *frame_range*:*slot_range*] [–**n** *node_range*]
[–**w** *host_name,host_name*, ...] [–**e** *host_name,host_name*, ...]
[–**N** *nodegroup,nodegroup*, ...] [–**a**] [–**G**] *dest_nodegroup*

Flags

–**h** Displays usage information.

–**s** Specifies a range of frames and slots on each frame to add to the node group.

–**n** Specifies a range of nodes to be added to the node group.

–**w** Specifies a comma-delimited list of hosts to add to the node group.

–**a** Specifies that all nodes in the current system partition be added to the node group. If the –**G** flag is also provided, all nodes in the SP system are included.

–**e** Specifies a comma-delimited exclusion list. These hosts are not added to the node group even if they are specified by another option.

–**N** Specifies a comma-delimited list of node groups to add to this node group.

–**G** Creates a global node group. System partition boundaries are ignored.

Operands

dest_nodegroup
Specifies the name associated with the node group being created.

Description

Use this command to create a node group named *dest_nodegroup*. The destination node group is populated based on the supplied options. Node group names must begin with a letter and can be followed by any letters or numbers, a period (.), or an underscore (_). If the destination node group already exists, you will receive an error.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to create and modify named node groups.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the "Managing Node Groups" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngcreate

Related Information

Commands: **ngaddto**, **ngdelete**, **ngdelfrom**, **ngfind**, **nglist**, **ngnew**, **ngresolve**

Examples

To create a node group called sample_ng that contains all the nodes in the current system partition except for k22n01, enter:

```
ngcreate -ae k22n01 sample_ng
```

ngdelete

Purpose

ngdelete – Removes node groups from persistent storage.

Syntax

ngdelete [–**h**] | [–**u**] [–**G**] *nodegroup* [*nodegroup* ...]

Flags

–**h** Displays usage information.

–**u** Removes the *nodegroup*, but leaves references to this *nodegroup* in the definition list of any any node group that contains it.

–**G** Specifies that the *nodegroup* is global.

Operands

nodegroup Specifies the name of the node group to be deleted.

Description

Use this command to remove node groups from persistent storage. By default, the node group is removed from any node group that contains it. If the –**u** flag is specified, references to this deleted node group will remain in containing node groups.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to delete node groups.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the “Managing Node Groups” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngdelete

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelfrom**, **ngfind**, **nglist**, **ngnew**, **ngresolve**

Examples

To delete nodegroups ngc and ngd, enter:

```
ngdelete ngc ngd
```

ngdelfrom

Purpose

ngdelfrom – Deletes nodes and node groups from the definition list of the destination node group.

Syntax

ngdelfrom [–**h**] | [–**G**] *dest_nodegroup*
nodenum | *nodegroup* [*nodenum* | *nodegroup*] ...

Flags

–**h** Displays usage information.

–**G** Specifies that the *dest_nodegroup* is global.

Operands

dest_nodegroup
Specifies the node group to be modified.

nodenum Specifies a node to remove. Nodes are specified as a space-delimited list of node numbers.

nodegroup Specifies a named node group to remove. Node groups are specified as a space-delimited list of node group names.

Note: Nodes numbers and node group names being removed can be intermixed.

Description

Use this command to remove nodes and node groups from the definition list of the destination node group. If the –**G** flag is specified, the *dest_nodegroup* must be global. If the –**G** flag is not specified, the *dest_nodegroup* must belong to the current system partition.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to modify named node groups.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the "Managing Node Groups" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngdelfrom

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelete**, **ngfind**, **nglist**, **ngnew**, **ngresolve**

Examples

To remove node 5 and node group ngc from nga, enter:

```
ngdelfrom nga 5 ngc
```

ngfind

Purpose

ngfind – Returns a list of all node groups whose definition list contains the specified node or node group.

Syntax

ngfind [–**h**] | [–**G**] *nodegroup* | *node*

Flags

–**h** Displays usage information.

–**G** Returns all global node groups that contain the specified global node group or node in their definition list. The default scope is the current system partition.

Operands

nodegroup Searches node group definition lists for references to this node group.

node Searches node group definition lists for references to this node.

Description

Use this command to list all node groups that contain the specified node or node group in their definition list. If the specified node group does not exist, you will receive an error. Use this command to determine what other node groups would be affected by changes to the specified node group.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the “Managing Node Groups” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngfind

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelete**, **ngdelfrom**, **nglist**, **ngnew**, **ngresolve**

Examples

To display a list of all node groups that contain node group test_B, enter:

```
> ngfind test_B
test_A
test_D
```

nglist

Purpose

nglist – Returns a list of all node groups in the current system partition.

Syntax

nglist [–**h**] | [–**G**]

Flags

–**h** Displays usage information.

–**G** Returns all global node groups.

Operands

None.

Description

Use this command to list all node groups in the current system partition to standard output. If the –**G** flag is specified, it will list all system node groups.

Standard Output

A list of node groups is written to standard output, one node group per line.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the "Managing Node Groups" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/nglist

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelete**, **ngdelfrom**, **ngfind**, **ngnew**, **ngresolve**

Examples

1. To display a list of all node groups in the current system partition, enter:

```
> nglist
nga
ngb
sampleng
test_A
```

2. To display a list of all global node groups, enter:

```
> nglist -G
g1
g2
g3
test_A
```

Note: The global node group test_A is not the same as node group test_A in the current system partition. The global scope and system partition dependent scope are independent name spaces and are stored in separate classes in the System Data Repository (SDR).

ngnew

Purpose

ngnew – Creates but does not populate new node groups in persistent storage.

Syntax

ngnew [–**h**] | [–**G**] *nodegroup* [*nodegroup* ...]

Flags

–**h** Displays usage information.

–**G** Creates global node groups.

Operands

nodegroup Specifies the node group to be created.

Description

Use this command to create new node groups. If the *nodegroup* already exists, you will receive an error. A valid node group name must begin with a letter. If the *nodegroup* is not a valid name, you will receive an error. If a node group in the list cannot be successfully created, it will not affect the creation of the other supplied node groups. A nonzero return code is returned.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to create persistent node groups.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the “Managing Node Groups” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngnew

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelete**, **ngdelfrom**, **ngfind**, **nglist**, **ngresolve**

Examples

To create node groups called nga, ngb, and ngc, enter:

```
ngnew nga ngb ngc
```

ngresolve

Purpose

ngresolve – Returns a list of hosts in the specified node group.

Syntax

ngresolve [–**h**] | [–**u** | –**n** | –**w** | –**d**] [–**G**] *nodegroup* [*nodegroup* ...]

Flags

–**h**	Displays usage information.
–**u**	Writes the definition list of *nodegroup*. Node groups contained by *nodegroup* are left unresolved.
–**n**	Specifies that nodes are written as node numbers. This is the default.
–**w**	Specifies that nodes are written as fully-qualified host names.
–**d**	Specifies that nodes are written as fully-qualified IP addresses.
–**G**	Specifies that node groups are global.

Operands

nodegroup	Specifies the node group to be resolved.

Description

Use this command to resolve the supplied named node groups into their constituent nodes. Nodes and node groups that are in the supplied node group but do not currently exist, will resolve to an empty list. If the –**u** flag is specified, these nonexistent nodes and node groups will be displayed.

Standard Output

A resolved list of nodes is written to standard output, one node per line.

Exit Values

0	Indicates the successful completion of the command.
nonzero	Indicates that an error occurred.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

Refer to the “Managing Node Groups” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional node grouping information.

Location

/usr/lpp/ssp/bin/ngresolve

Related Information

Commands: **ngaddto**, **ngcreate**, **ngdelete**, **ngdelfrom**, **ngfind nglist**, **ngnew**

Examples

1. To display the definition list for node group nga, enter:

```
> ngresolve -u nga
1
3
ngb
```

2. To resolve node group nga into its constituent nodes, enter:

```
> ngresolve nga
1
3
6
8
```

3. To resolve node group nga into fully-qualified host names, enter:

```
> ngresolve -w nga
k22n01.ppd.pok.ibm.com
k22n03.ppd.pok.ibm.com
k22n06.ppd.pok.ibm.com
k22n08.ppd.pok.ibm.com
```

4. To display the IP addresses of the nodes in node group nga, enter:

```
> ngresolve -d nga
129.40.157.65
129.40.157.67
129.40.157.70
129.40.157.72
```

nodecond

Purpose

nodecond – Conditions an SP processing node.

Syntax

nodecond [–**G**] [–**n**] *frame_ID slot_ID*

Flags

–**G** Specifies Global mode. With this flag, the node to be conditioned can be outside of the current system partition.

–**n** Obtains the Ethernet hardware address instead of doing a network boot.

Operands

frame_ID Specifies the number of the frame containing the node to be conditioned.

slot_ID Specifies the number of the slot containing the node to be conditioned.

Description

Node conditioning is the administrative procedure used to obtain the Ethernet hardware address of an SP processing node or to initiate a network boot of an SP processing node. The Ethernet hardware address is required by SP System Management for the proper configuration of the system. A network boot of the node is required by the System Management installation procedures.

By default, the **nodecond** command initiates a network boot of the node specified by the *frame_ID* and *slot_ID* operands. The specified node must be in the current system partition unless the –**G** flag is also specified. The frame ID is any configured frame number and the slot ID is taken from the set 1 through 16. The command completes when the node has booted to the point of configuring its console. Optionally, the **nodecond** command obtains the Ethernet hardware address of the processing node, specified by the *frame_ID* and *slot_ID* operands. The hardware address is written to standard output and the node is left powered off with the keylock in the Normal position.

As the command executes, it writes status information indicating its progress to /**var**/**adm**/**SPlogs**/**spmon**/**nc**/**nc.***frame_ID.slot_ID*.

This command uses the SP Hardware Monitor. Therefore, the user must be authorized to access the Hardware Monitor subsystem and, for the frame specified to the command, the user must be granted Virtual Front Operator Panel (VFOP) and S1 (serial port on the node that you can access via the **s1term** command) permission. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **kinit** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

Files

/usr/lpp/ssp/bin/nodecond
Contains the **nodecond** command.

/var/adm/SPlogs/spmon/nc
Directory containing **nodecond** status files.

Related Information

Commands: **hmcmds**, **hmmon**, **s1term**

Examples

1. To fetch the Ethernet hardware address of the node in frame 5 in slot 1 and save it in a file, enter:

   ```
   nodecond -n 5 1 > eth_adrr.5.1
   ```

2. To network boot the node in frame 7 in slot 16, enter:

   ```
   nodecond 7 16
   ```

nrunacct

Purpose

nrunacct – Runs on each node every night to merge raw accounting data from the login, fee, disk, print, and process subsystems.

Syntax

nrunacct *yyyymmdd*
[**SETUP** | **WTMPFIX** | **CONNECT1** | **CONNECT2** | **PROCESS** | **MERGE** | **FEES** | **DISK** | **QUEUEACCT** | **CMS** | **USEREXIT** | **CLEANUP**]

Flags

SETUP Moves the active accounting files to working files and restarts the active files.

WTMPFIX Verifies the integrity of the **wtmp** file and corrects dates if necessary.

CONNECT1 Calls the **acctcon1** command to produce connect session records.

CONNECT2 Converts connect session records into total accounting records (**tacct.h** format).

PROCESS Converts process accounting records into total accounting records (**tacct.h** format). Filters out the records that belong to processes that were part of a job that had exclusive use of the node and appends a total accounting fee record to the fee file for each of these jobs. Records are identified as belonging to processes that were part of a job that had exclusive use of the node, only if exclusive use accounting was enabled at the time the job ran.

MERGE Merges the connect and process total accounting records.

FEES Converts accounting fee file records into total accounting records (**tacct.h** format) and merges them with the connect and process total accounting records.

DISK Merges disk accounting records with connect, process, and fee total accounting records.

QUEUEACCT
Sorts the queue (printer) accounting records, converts them into total accounting records (**tacct.h** format), and merges them with other total accounting records.

CMS Produces command summaries and updates the file that records the date each user last logged into the node.

USEREXIT If the **/var/adm/nsiteacct** shell file exists, calls it at this point to perform site-dependent processing.

CLEANUP Deletes temporary files and exits.

Operands

yyyymmdd Specifies the date when accounting is to be rerun.

Description

The **nrunacct** command is the main daily accounting shell procedure, for each individual node. Normally initiated by the **cron** daemon, the **nrunacct** command merges the day's raw connect, fee, disk, queuing system (printer), and process accounting data files for the node.

This command has two parameters that must be entered from the keyboard should you need to restart the **nrunacct** procedure. The date parameter, *YYYYMMDD* enables you to specify the date for which you want to rerun the node accounting. The state parameter enables a user with administrative authority to restart the **nrunacct** procedure at any of its states. For more information on restarting **nrunacct** procedures and on recovering from failures, see "Restart Procedure."

The **nrunacct** command protects active accounting files and summary files in the event of runtime errors, and records its progress by writing descriptive messages into the **/var/adm/acct/nite/active***YYYYMMDD* file. When the **nrunacct** procedure encounters an error, it sends mail to users root and adm, and writes standard errors to **/var/adm/acct/nite/accterr**.

The **nrunacct** procedure also creates two temporary files, **lock** and **lock1**, in the directory **/var/adm/acct/nite**, which it uses to prevent two simultaneous calls to the **nrunacct** procedure. It uses the lastdate file (in the same directory) to prevent more than one invocation per day.

The **nrunacct** command breaks its processing into separate, restartable states. As it completes each state, it writes the name of the next state in the **/var/adm/acct/nite/state***YYYYMMDD* file.

Restart Procedure

To restart the **nrunacct** command after a failure, first check the **/var/adm/acct/nite/active***YYYYMMDD* file for diagnostic messages, then fix any damaged data files, such as **pacct** or **wtmp**. Remove the lock files and **lastdate** file (all in the **/var/adm/acct/nite** directory, before restarting the **nrunacct** command. You must specify the *YYYYMMDD* parameter if you are restarting the **nrunacct** command. It specifies the date for which the **nrunacct** command is to rerun accounting. The **nrunacct** procedure determines the entry point for processing by reading the **/var/adm/acct/nite/statefile***YYYYMMDD* file. To override this default action, specify the desired state on the **nrunacct** command line.

It is not usually a good idea to restart the **nrunacct** command in the SETUP state. Instead, perform the setup actions manually and restart accounting with the WTMPFIX state, as follows:

```
/usr/lpp/ssp/bin/nrunacct YYYYMMDD WTMPFIX
```

If the nrunacct command fails in the PROCESS state, remove the last **ptacct** file, because it is incomplete.

Files

/var/adm/wtmp
Log in/log off history file.

/var/adm/pacct*
Process accounting file.

/var/adm/acct/nite/dacct
Disk usage accounting file.

/var/adm/qacct
Active queue accounting file.

/var/adm/fee Record of fees charged to users.

/var/adm/acct/nite/ptacct*.mmdd
Summary version of pacct files.

/var/adm/acct/nite/activeYYYYMMDD
The nrunacct message file.

/var/adm/acct/nite/lock*
Prevents simultaneous invocation of **nrunacct**.

/var/adm/acct/nite/lastdate
Contains last date nrunacct was run.

/var/adm/acct/nite/statefileYYYYMMDD
Contains current state to process.

Restrictions

Access Control: This command should grant execute (x) access only to members of the adm group.

Related Information

Commands: **acctcms**, **acctcom**, **acctcon1**, **acctcon2**, **acctmerg**, **accton**, **acctprc1**, **acctprc2**, **crontab**, **fwtmp**, **nrunacct**,

Daemon: **cron**

Subroutine: **acct**

File format: **acct**, **failedlogin**, **tacct**, **wtmp**

The System Accounting information found in *AIX Version 4 System Management Guide*.

Examples

1. To restart a node's system accounting procedures for a specific date, enter a command similar to the following:

   ```
   nohup /usr/lpp/ssp/bin/nrunacct 19950601 2>> \
   /var/adm/acct/nite/accterr &
   ```

   This example restarts **nrunacct** for the day of June 1 (0601), 1995. The **nrunacct** command reads the file **/var/adm/acct/nite/statefile19950601** to find out the state with which to begin. The **nrunacct** command runs in the

background (&), ignoring all INTERRUPT and QUIT signals (**nohup**). Standard error output (2) is added to the end (>>) of the /**var**/**adm**/**acct**/**nite**/**accterr** file.

2. To restart a node's system accounting procedures for a particular date at a specific state, enter a command similar to the following

   ```
   nohup /usr/lpp/ssp/bin/nrunacct 19950601 FEES 2>> \
   /var/adm/acct/nite/accterr &
   ```

   This example restarts the **nrunacct** command for the day of June 1 (0601), 1995, starting with the FEES state. The **nrunacct** command runs in the background (&), ignoring all INTERRUPT and QUIT signals (the **nohup** command). Standard error output (2) is added to the end (>>) of the /**var**/**adm**/**acct**/**nite**/**accterr** file.

p_cat

Purpose

p_cat – Issues a parallel cat of files.

Syntax

p_cat [–**w** – | *noderange* | '*hostlist args*'] *file_name file_name* ...

Parameters

The **p_cat** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is valid only for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

file_name Specifies a list of file names on the hosts to be concatenated to standard output.

Description

The **p_cat** command issues the AIX **cat** command on multiple hosts. **p_cat** uses **dsh** to execute the **cat** command on multiple hosts. The output of the **cat** command is written to standard output.

Since **p_cat** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

SP commands: **dsh**, **pexec**

AIX command: **cat**

Examples

To copy /**.rhosts** from each host1, host2, and host3 to the local /**.rhosts** file (described previously), enter:

```
p_cat -w host1,host2,host3 /.rhosts >> /.rhosts
```

pcp

Purpose

pcp – Specifies a parallel copy of local files and directories to other hosts.

Syntax

pcp [–**w** – | *noderange* | '*hostlist args*'] [–**p**] [–**r**]
localfile_or_dir remotefile_or_dir

Parameters

The **pcp** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

–**p** Preserves the modification times and modes of the source files in the copies sent to the destination only if the user has root authority or is the owner of the destination. Without this flag, the **umask** command at the destination modifies the mode of the destination file, and the modification time of the destination file is set to the time the file is received.

–**r** Recursively copies subtrees for directories.

local_file_or_dir
Contains the name of the file or directory to be copied to the remote hosts. If a full path name is not specified, the file is looked for relative to the user's current working directory.

remote_file_or_dir
Contains the name of the file or directory to which the file or directory is copied.

Description

The **pcp** command copies files from the local host to one or more others in parallel. **pcp** is similar to **rcp** and in fact uses SP **rcp** via **dsh**. The –**r** and –**p** flags are passed to SP **rcp**.

Since **pcp** uses **dsh** and SP **rcp**, proper authentication and authorization to issue these commands is necessary.

Note: Since the **pcp** command uses the secure version of **rcp**, your **.klogin** or **.rhosts** files need to be set up to authorize you on each of the nodes to which you are copying a file. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide*. Otherwise, you see:

```
Permission denied
```

messages from the nodes for which you are not authorized.

Related Information

Commands: **dsh**, **hostlist**

SP Command: **rcp**

Examples

1. To copy a local file to host1 and host2 and rename it on those hosts, enter:

   ```
   pcp -w host1,host2 /etc/fileq /etc/filen
   ```

2. To copy a file in the current directory to a particular directory on all the nodes in the SP system, enter:

   ```
   pcp '-a' sysctl.acl /etc
   ```

3. To copy a directory subtree to all the hosts in the SP system that are currently responding, except for "badnode," enter:

   ```
   hostlist -av -e badnode | pcp -w - -r /etc/new /etc/new
   ```

4. To copy a directory subtree to all the hosts in the SP system that are currently responding, except for "badnode," enter:

   ```
   pcp "-av -e badnode" -r /etc/new /etc/new
   ```

pdf

Purpose

pdf – Displays file system statistics on multiple nodes in parallel.

Syntax

pdf [–**w** – | *noderange* | '*hostlist args*'] [*file system1* [*file system2* ...]]

Parameters

The **pdf** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

Acceptable parameters to the **pdf** command are zero or more blank-separated file system names to be displayed. If no parameters are supplied, all local file systems (on each specified node) are displayed.

Description

The **pdf** command displays file systems and their usage statistics on one or more nodes in parallel. **pdf** is similar to the **df** command, but does not use it. Differences are:

- The **pdf** command can be executed on more than one node at a time.
- Execution is authorized to any authenticated user (this can be changed by the system administrator).
- The **pdf** command output contains more information than that of the **df** command.

Since **pdf** uses **sysctl**, proper authentication and authorization to issue these commands is necessary.

Related Information

Commands: **hostlist**, **sysctl**

Examples

1. To list all file systems and their usage statistics on all hosts in the SP system, enter:

   ```
   pdf '-a'
   ```

2. To list the usage statistics for the /**tmp** file system on nodes named node1 and node2, enter:

   ```
   pdf -w node1,node2 /tmp
   ```

penotify

Purpose

penotify – Adds, removes, or shows AIX error log notification objects.

Syntax

penotify [–**a**] [–**c** *class*] [–**f add**] [–**l** *label*] [–**m** *method*] [–**n** *name*] [–**p** *pid*] [–**t** *type*] [–**w** *hosts*] [–**A** *alert*] [–**C** *rclass*] [–**N** *rname*] [–**P**] [–**T** *rtype*]

penotify [–**a**] [–**f remove**] [–**w** *hosts*] [–**n** *name*]

penotify [–**a**] [–**f show**] [–**w** *hosts*] [–**n** *name*]

Flags

–**a**	Executes on all nodes in the system partition.
–**c** *class*	Specifies the error class.
–**f** *func*	Specifies the function: **add**, **remove**, or **show**.
–**h**	Displays usage information.
–**l** *label*	Specifies the error label.
–**m** *method*	Specifies the notification method.
–**n** *name*	Specifies the name of the notification object.
–**p** *pid*	Specifies the process ID for the notification object.
–**t** *type*	Specifies the error type.
–**w** *hosts*	Runs the command on a file or a list of host names.
–**A** *alert*	Specifies the match alertable errors (**true** or **false**).
–**C** *rclass*	Specifies the resource class.
–**N** *rname*	Specifies the resource name.
–**P**	Specifies whether to persist across system restart (**yes**, if –**P** is provided).
–**T** *rtype*	Specifies the resource type.

Operands

None.

Description

Use this command to add, remove, or show notification objects in the errnotify Object Data Management (ODM) class. The AIX **errdemon** matches logged errors to objects in this class to execute a method defined in the class object. The error class, error label, error type, alert, resource name, resource class, and resource type parameters are used for matching to logged errors. Refer to the *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for descriptions of error notification object class fields.

When a match occurs, the **errdemon** executes the notify method passing up to nine ($1–$9) parameters related to the logged error.

If the –**w** parameter begins with a slash (/), it is interpreted as a file containing a list of nodes to execute the command on; otherwise, it can be a comma-delimited list of host names or a single-quoted, blank-delimited list of host names. If neither the –**a** nor –**w** parameters are used, the command defaults to the local node.

Security

You must have a Kerberos principal defined in the /**etc/logmgt.acl** file to run this command.

Related Information

The *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

The *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

1. To view all notification objects on nodes k4710, k4712, and k4715, enter:

   ```
   penotify -w k47n10,k47n12,k47n15 -f show
   ```

2. To remove the notification object named HDISK0_ERR on all nodes, enter:

   ```
   penotify -a -f remove -n HDISK0_ERR
   ```

3. To add a notification object to the nodes in the /**tmp**/**nodelist** file, enter:

   ```
   penotify -w /tmp/nodelist -f add -n PEND_ERR -P \
   -m'/spdata/sys1/EN_meth/EN_pend $1' -t PEND -c S
   ```

   This adds a notification object named PEND_ERR to all nodes in the /**tmp**/**nodelist** file. The object will **persist** when the system is restarted, and will match error records of type PEND and class S. The method that is executed by **errdemon** when a matching error occurs will be /**spdata**/**sys1**/**EN_meth**/**EN_pend**, and it will be passed the $1 parameter (sequence number). The notification method must be accessible to each node.

perspectives

Purpose

perspectives – Invokes the launch pad of the SP Perspectives graphical user interface (GUI).

Syntax

perspectives [–**userProfile** *name*] [–**systemProfile** *name*]
[–**noProfile**] [{–**backgroundColor** | **bg**} *colorName*]
[{–**foregroundColor** | **fg**} *colorName*]
[–**fontFamily** *name*] [–**fontSize** *size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides any foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to “Perspectives Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to invoke the SP Perspectives Launch Pad. The Launch Pad is a small, customizable GUI from which the user can start (or launch) executables associated with maintaining and monitoring an IBM RS/6000 SP.

The mail window shows an icon for each executable that can be launched. Double-clicking on an icon launches the associated executable.

Preferences that define the look and layout of the Perspectives window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

The User's Preferences are read from and saved to **$HOME/.perspectives(User Profile Name)**.

The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.perspectives(System Profile name)**.

Restrictions

Any user may run perspectives. Launching certain executables may require root privilege to run.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For information on using the perspectives GUI, please see the online help and the "Using the SP Perspectives GUI" chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/perspectives

Related Information

Specific perspective windows may be brought up directly by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, **spsyspar**, and **spvsd**.

Examples

1. To invoke the perspectives launch pad, enter:

   ```
   perspectives
   ```

2. To run force perspectives to display a 14 point type regardless of what is set in the preference files, enter:

   ```
   perspectives -fontSize 14
   ```

pexec

Purpose

pexec – Specifies the parallel execution of a command.

Syntax

pexec [–**w** – | *noderange* | '*hostlist args*'] *command command_args*

Parameters

The **pexec** commands requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

command Specifies a command to execute on the hosts in parallel.

cmd_args Specifies arguments to the command.

Description

The **pexec** command issues a command on multiple hosts in parallel. The output is formatted so that distinct output is displayed only once. The **pexec** command uses **dsh** to execute the specified command on multiple hosts. The output of the **ls** command is written to standard output and formatted. The **pls**, **prm**, **pmv**, **pfind**, and **pps** commands are simply links to **pexec**.

Since **pexec** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Note: If any of the **pls**, **prm**, **pfind**, **pps**, **pmv** are renamed, they do not work properly.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **dshbak**, **hostlist**

Examples

1. To list the contents of /**usr** from each host1, host2, and host3 (described previously), enter:

   ```
   pexec -w host1,host2,host3 ls /
   ```

2. To copy a directory subtree to all the hosts in the SP system that are currently responding, except for "badnode," enter:

   ```
   hostlist -a -v -e badnode | pexec -w -cp -r /etc/new /etc/new
   ```

3. Another way to enter the command in the previous example follows:

   ```
   pexec "-a -v -e badnode" cp -r /etc/new /etc/new
   ```

pexscr

Purpose

pexscr – Runs local and remote programs in parallel.

Syntax

pexscr

Flags

None.

Operands

None.

Description

The **pexscr** command executes particular commands on particular processors in parallel. **pexscr** reads lines of the following format from standard input:

```
host_name: arbitrary_command
```

and executes each *arbitrary_command* on the specified host. All commands are run in parallel. SP **rsh** is used to run the remote commands, and local commands are run directly. Host names can include any parameter that may be specified on the **rsh** command.

Since **pexscr** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Related Information

Commands: **dsh**, **rsh**

Examples

To remove a file on host1 and rename a file on host2 simultaneously, enter:

```
pexscr <<END
host1: rm /tmp/shnozzola
host2: mv /tmp/shnozzola /tmp/bozo
END
```

pfck

Purpose

pfck – Displays file system statistics on multiple hosts in parallel based on usage criteria.

Syntax

pfck [–**w** – | *noderange* | '*hostlist args*'] { [–**s** *num*] [–**f** *num*] [–**u** *num*] [–**pf** *num*] [–**pu** *num*] [–**is** *num*] [–**if** *num*] [–**iu** *num*] [–**pif** *num*] [–**piu** *num*] }

Parameters

The **pfck** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2 and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

Subsequent flags specify file system usage statistics criteria to be applied in searching for file systems to display. At least one of the following flags must be specified. Multiple flags are allowed.

–**s** *num* Indicates that the file system size is > *num* kilobytes (KB).
–**f** *num* Indicates that the file system free space is < *num* kilobytes (KB).
–**u** *num* Indicates that the file system used space is > *num* kilobytes (KB).
–**pf** *num* Indicates that the file system free space is < *num* %.
–**pu** *num* Indicates that the file system used space is > *num* %.
–**is** *num* Indicates that the file system inodes are > *num*.
–**if** *num* Indicates that the file system free inodes are < *num*.
–**iu** *num* Indicates that the file system used inodes are > *num*.
–**pif** *num* Indicates that the file system free inodes are < *num* %.
–**piu** *num* Indicates that the file system used inodes are > *num* %.

File system usage statistics criteria are logically **OR**ed together when comparing against actual usage information. That is, if a file system meets *any* of the search criteria, then it is displayed.

Description

The **pfck** command displays file systems and their usage statistics from one or more nodes in parallel based on usage criteria.

Since **pfck** uses **sysctl**, proper authentication and authorization to issue these commands is necessary.

Related Information

Commands: **hostlist**, **sysctl**

Examples

1. To list all file systems with less than 20% free space on all nodes in the SP system, enter:

   ```
   pfck -a -pf 20
   ```

2. To list all file systems on the nodes named node1, node2, and node4 which are greater than 98% full, enter:

   ```
   pfck -w node1,node2,node4 -pu 98
   ```

pfind

Purpose

pfind – Specifies a parallel find of files with a matching expression.

Syntax

pfind [–**w** – | *noderange* | '*hostlist args*'] *find_args*

Parameters

The **pfind** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

find_args Specifies arguments to the AIX **find** command.

Description

The **pfind** command issues the AIX **find** command on multiple hosts. The output is formatted so that distinct output is displayed only once. The **pfind** command uses **dsh** to execute the **find** command on multiple hosts. The output of the **ls** commands is written to standard output and formatted. The **pfind** command is identical to **pexec find**.

Since **pfind** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **find**, **hostlist**, **pexec**

Examples

To find out if the file **elvis** is contained in /**usr**/**bin** on any host1, host2, and host3 (described previously), enter:

```
pfind -w host1,host2,host3 /usr/bin -print -name "elvis"
```

pfps

Purpose

pfps – Finds and performs operations on processes on multiple hosts in parallel based on the value of an expression.

Syntax

pfps [–**w** – | *noderange* | '*hostlist args*'] *operation* [*expression*]

Parameters

The **pfps** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

operation Must be at least one of the following:

–**print** If expression is not specified, all process information is listed. If expression is specified, process information for which the expression is true is listed. Output is in **ps auw** format.

–**kill** *signal* Causes matching processes to be killed with the specified signal if the user is authorized or owns the process. The signal can be specified as a number or the name of the signal (for example, HUP).

–**nice** *n* Causes the nice value to be set for matching processes if the user is authorized.

expression The *expression* must be at least one of the following:

–**n** *name* Evaluates to true if the fully qualified name of the process matches the name.

–**tn** *name* Evaluates to true if the name of the process matches the name.

–**o** *owner* Evaluates to true if the user name of the process matches the owner.

–**pty** *name* Evaluates to true if the name matches the process controlling terminal.

–**rtime** *hh:mm*
Evaluates to true for processes whose total execution time is *hh:mm* time or longer.

–**stime** *dd:hh:mm*
Evaluates to true for processes that have started at least *dd* days, *hh* hours, and *mm* minutes ago.

–**r** *state* Evaluates to true for processes in the specified run state.

–**cpu** *percentage*
Evaluates to true for processes using greater than the specified percentage of system CPU.

–**mem** *percentage*
Evaluates to true for processes consuming more than the specified percentage of system memory.

–**or** Evaluates the expression by **OR**ing together the terms.

Description

The **pfps** command performs operations on processes on one or more hosts in parallel. These operations include printing information about processes (–**print**), sending a signal to the process (–**kill**), and changing the priority of the process (–**nice**).

Authorization is via an Access Control List (ACL) on each node and is required when users try to **kill** a process that they do not own or **nice** a process to a higher priority. ACLs for **pfs** are contained in the /**etc**/**sysctl.pfps.acl** file.

An expression can also be specified using the preceding flags to select processes for when the expression evaluates to true. Flags are **AND**ed together unless the –**or** flag is used.

Parentheses can be used to group flags, but parenthesis must be separated from flags by a space. Also, parenthesis or any special shell character should be escaped with a backslash (\).

Since **pfps** uses **sysctl**, proper authentication and authorization to issue these commands is necessary.

Files

/etc/sysctl.pfps.acl
The ACL file which authorizes listed principals to use the **nice** and **kill** options.

Related Information

Commands: **hostlist**, **kill**, **nice**, **ps**, **renice**, **sysctl**

Examples

1. To list all processes on all hosts in the SP system, enter:

   ```
   pfps '-a' -print
   ```

2. To restart **daemon** processes on host1 and host2 that were running for more than one day (the user must be listed in the **/etc/sysctl.pfps.acl** on each host or the command is ignored for that host), enter:

   ```
   pfps -w host1,host2 -rtime 24:00 -tn daemond -kill HUP
   ```

3. To list all processes belonging to root that are using more than 10% of system CPU or 10% of system memory on hosts listed in the **./wcollective** file, enter:

   ```
   WCOLL=./wcollective pfps '' \( -cpu 10 -or -mem 10 \) -o root -print
   ```

pls

Purpose

pls – Specifies a parallel list of files and directories.

Syntax

pls [–**w** – | *noderange* | '*hostlist args*'] *ls_args*

Parameters

The **pls** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

ls_args Specifies arguments to the AIX **ls** command.

Description

The **pls** command issues the AIX **ls** command on multiple hosts. The output is formatted so that duplicate output is displayed only once. The **pls** command uses **dsh** to execute the **ls** command on multiple hosts. The output of the **ls** commands is written to standard output and formatted. The **pls** command is identical to **pexec ls**.

Since **pls** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **ls**, **pexec**

Examples

To list the contents of /**usr** from each host1, host2, and host3 (described previously), enter:

```
pls -w host1,host2,host3 /usr
```

pmanctrl

Purpose

pmanctrl – Controls the Problem Management subsystem.

Syntax

pmanctrl {–**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h**}

Flags

–**a**	Adds the Problem Management subsystem.
–**s**	Starts the Problem Management subsystem.
–**k**	Stops the Problem Management subsystem.
–**d**	Deletes the Problem Management subsystem.
–**c**	Deletes the Problem Management subsystem from all system partitions.
–**t**	Turns tracing on for the Problem Management subsystem.
–**o**	Turns tracing off for the Problem Management subsystem.
–**r**	Required by the **syspar_ctrl** command, but does not do anything.
–**h**	Displays usage information.

Operands

None.

Description

Problem Management is a general purpose facility for monitoring and reacting to specific event occurrences within the SP system. The **pmanctrl** command controls the operations of the subsystems that are required for Problem Management. These subsystems are under the control of the System Resource Controller (SRC) and belong to a subsystem group called **pman**. Associated with each subsystem is a daemon.

An instance of the Problem Management subsystem executes on the control workstation and on every node of a system partition. Because Problem Management provides its services within the scope of a system partition, its subsystems are said to be system partition sensitive. For this reason, the **pmanctrl** command is normally invoked by the **syspar_ctrl** command during installation of the system, boot or reboot of individual nodes, and partitioning or repartitioning of the system.

From an operational point of view, the Problem Management subsystem group is organized as follows:

Subsystem: Problem Management

Subsystem Group: pman

SRC subsystems: pman and **pmanrm**

The **pman** subsystem is associated with the **pmand** daemon. The **pmanrm** subsystem is associated with the **pmanrmd** daemon.

The subsystem names on the nodes are **pman** and **pmanrm**. There is one of each subsystem per node and it is associated with the system partition to which the node belongs.

On the control workstation, there are multiple instances of each subsystem, one for each system partition. Accordingly, the subsystem names on the control workstation have the system partition name appended to them. For example, for system partitions named **sp_prod** and **sp_test**, the subsystems on the control workstation are named **pman.sp_prod**, **pman.sp_test**, **pmanrm.sp_prod**, and **pmanrm.sp_test**.

Daemons: **pmand** and **pmanrmd**

The **pmand** daemon provides the majority of Problem Management functions.

The **pmanrmd** daemon provides command-based resource monitor data to the **pmand** daemon.

The **pmanctrl** command provides a variety of controls for operating the Problem Management subsystems:

- Adding, starting, stopping, and deleting the subsystems
- Cleaning up the subsystems, that is, deleting them from all system partitions
- Turning tracing on and off

Unless the **–c** flag is used, the **pmanctrl** command only operates within a single partition. On a node, the **pmanctrl** command operates within the system partition to which the node belongs. On the control workstation, the **pmanctrl** command operates within any single partition, which can be chosen by setting the SP_NAME environment variable.

When the **pmanctrl** command is called with the **–a** flag, it uses the **mkssys** command to add the subsystems to the SRC, and it takes the necessary steps to make sure that the subsystems are automatically started when the node is booted.

When the **pmanctrl** command is called with the **–s** flag, it uses the **startsrc** command to start the **pman** and **pmanrm** subsystems.

When the **pmanctrl** command is called with the **–k** flag, it uses the **stopsrc** command to stop the **pman** and **pmanrm** subsystems.

When the **pmanctrl** command is called with the **–d** flag, it uses the **rmssys** command to delete the subsystems from the SRC, and if there are no more Problem Management subsystems remaining, it makes sure there is no **/etc/inittab** entry for the Problem Management subsystem.

When the **pmanctrl** command is called with the **–c** flag, it stops all running Problem Management subsystems, removes them all from the SRC, and makes sure there is no **/etc/inittab** entry for the Problem Management subsystem.

When the **pmanctrl** command is called with the –**t** flag, it uses the **traceson** command to turn on tracing in the **pman** subsystem. Tracing is not available for the **pmanrmd** subsystem.

When the **pmanctrl** command is called with the –**o** flag, it uses the **tracesoff** command to turn off tracing in the **pman** subsystem. Tracing is not available for the **pmanrmd** subsystem.

While they are running, the Problem Management daemons provide information about their operation and errors by writing entries in a log file that is located in the **/var/adm/SPlogs/pman** directory. On the control workstation, the **pmand** daemon writes to a log file named **pmand.***syspar_name***.log**, and the **pmanrmd** daemon writes to a log file named **pmanrmd.***syspar_name***.log**, where *syspar_name* is the name of the system partition. On the nodes, the **pmand** daemon writes to a log file named **pmand.log** and the **pmanrmd** daemon writes to a log file named **pmanrmd.log**.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

The “Using the Problem Management Subsystem” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*

IBM AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *IBM AIX Version 4 General Programming Concepts: Writing and Debugging Programs*

Location

/usr/lpp/ssp/bin/pmanctrl

Related Information

SP Command: **syspar_ctrl**

AIX Commands: **mkssys**, **rmssys**, **startsrc**, **stopsrc**

Examples

1. To add the Problem Management subsystem to the SRC, enter:

   ```
   pmanctrl -a
   ```

2. To start the Problem Management subsystem, enter:

   ```
   pmanctrl -s
   ```

3. To stop the Problem Management subsystem, enter:

   ```
   pmanctrl -k
   ```

4. To delete the Problem Management subsystem from the SRC, enter:

   ```
   pmanctrl -d
   ```

5. To clean up the Problem Management subsystem on all system partitions, enter:

   ```
   pmanctrl -c
   ```

6. To turn tracing on for the Problem Management daemon, enter:

   ```
   pmanctrl -t
   ```

7. To turn tracing off for the Problem Management daemon, enter:

   ```
   pmanctrl -o
   ```

pmandef

Purpose

pmandef – Defines events and resulting actions to the Problem Management subsystem.

Syntax

To subscribe to an event and associate a set of actions with that event, use the following:

pmandef –**s** *HandleName* –**e** *ResourceVariable:InstanceVector:Predicate*
[–**r** *RearmPredicate*] [–**c** *EventCommand*] [–**C** *RearmCommand*]
[–**t** *EventTrapid*] [–**T** *RearmTrapid*] [–**l** *EventLogText*]
[–**L** *RearmLogText*] [–**x** *EventCmdTimeout*]
[–**X** *RearmCmdTimeout*] [–**U** *UserName*] [–**m** *UserLabel*]
[–**h** {*Host1*[,*Host2*,...] | – | **local**} | –**N** *NodeGroup* | –**n** *NodeRange*]

To deactivate or activate a Problem Management subscription, use the following:

pmandef {–**d** | –**a**} {*HandleName* | **all**}
[–**h** {*Host1*[,*Host2*,...] | – | **local**} | –**N** *NodeGroup* | –**n** *NodeRange*]

To query or remove a Problem Management subscription, use the following:

pmandef {–**q** | –**u**} {*HandleName* | **all**}

Flags

Flags that specify the type of request follow:

–**s** *HandleName*
Specifies that this is a subscribe request and the remaining flags define the Problem Management subscription. The *HandleName* provides a means to identify this subscription to Problem Management using the –**d**, –**a**, –**u**, or –**q** flags. The **all** keyword cannot be used as a handle name for a subscribe request.

–**d {***HandleName* | **all}**
Specifies that the actions associated with the subscription identified by *HandleName* should be turned off or deactivated. The **all** keyword deactivates all subscriptions owned by the caller's Kerberos principal.

–**a {***HandleName* | **all}**
Specifies that the actions associated with the subscription identified by *HandleName* should be turned on or activated. The **all** keyword activates all subscriptions owned by the caller's Kerberos principal.

–**u {***HandleName* | **all}**
Specifies that this is an unsubscribe request and the subscription identified by *HandleName* should be removed. The **all** keyword unsubscribes all subscriptions owned by the caller's Kerberos principal.

–**q {***HandleName* | **all}**
Requests all of the Problem Management daemons, for which the subscription identified by *HandleName* is defined, to provide status

about the named subscription. The **all** keyword queries all subscriptions owned by the caller's Kerberos principal.

Flags that specify the hosts to be affected by the request follow:

–h {*Host1***[,***Host2***,...]** | – | **local}**
For a subscribe request, specifies the hosts that belong to the subscription. For an activate or deactivate request, specifies the hosts to be activated or deactivated, which can only include hosts that belong to the subscription. The hosts may be specified as a comma-separated list of host names or the hyphen (-) may be used to indicate that host names are to be read from standard input, or the **local** keyword may be used to indicate that the list of hosts is to be obtained from the *NodeNum* field in the event instance vector. If the **local** keyword is specified, the *NodeNum* field cannot contain wildcards. Use of the **local** keyword also causes any resulting actions to occur on the same host where the event occurs. All specified hosts must reside in the same system partition.

–N *NodeGroup*
For a subscribe request, specifies the node group that contains all of the hosts that belong to the subscription. For an activate or deactivate request, specifies the node group that contains all of the hosts to be activated or deactivated, which can only include hosts that belong to the subscription. All specified hosts must be in a partitioned-bound node group.

–n *NodeRange*
For a subscribe request, specifies the node range that contains all of the hosts that belong to the subscription. For an activate or deactivate request, specifies the node range that contains all of the hosts to be activated or deactivated, which can only include hosts that belong to the subscription. A node range is a series of numbers that are separated by commas and hyphens, which indicate a set of node numbers, such as 0—3, 5, 8—10. All specified hosts must reside in the same system partition.

Flags that define a Problem Management subscription follow:

–e *ResourceVariable***:***InstanceVector***:***Predicate*
Specifies the Event Management resource variable, instance vector and predicate that define the event for which actions are generated for this Problem Management subscription. Refer to *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference* for further information about Event Management event definitions.

–r *RearmPredicate*
Specifies the Event Management re-arm predicate, which together with the resource variable, instance vector and predicate specified by the **–e** flag, defines the re-arm event for which actions are generated for this Problem Management subscription. Refer to *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference* for further information about Event Management event definitions.

–c *EventCommand*
Specifies a command to be executed when the event defined by the **–e** flag occurs. The command will be interpreted by the user's login program, so *EventCommand* may contain additional arguments and shell metacharacters. For example:

```
echo this is a test >/tmp/event.out
```

is allowed.

–C *RearmCommand*
Specifies a command to be executed when the re-arm event defined by the –r flag occurs. The command will be interpreted by the user's login program, so *RearmCommand* may contain additional arguments and shell metacharacters. For example:

```
echo this is a test >/tmp/rearm.out
```

is allowed.

–t *EventTrapid*
Specifies that a Simple Network Management Protocol (SNMP) trap should be sent when the event defined by the **–e** flag occurs. *EventTrapid* is the specific trap ID to be used.

–T *RearmTrapid*
Specifies that an SNMP trap should be sent when the re-arm event defined by the **–r** flag occurs. *RearmTrapid* is the specific trap ID to be used.

–l *EventLogText*
Specifies text that should be written to the AIX error log and BSD syslog when the event defined by the **–e** flag occurs.

–L *RearmLogText*
Specifies text that should be written to the AIX error log and BSD syslog when the re-arm event defined by the **–r** flag occurs.

–x *EventCmdTimeout*
Specifies a time limit in seconds for the command specified by the **–c** flag. If the command does not complete within *EventCmdTimeout* seconds, it will be sent a SIGTERM signal. If the command does not terminate after an additional 5 seconds, the SIGTERM signal will be followed by a SIGKILL signal. If the **–x** flag is not specified, the command runs until completion.

–X *RearmCmdTimeout*
Specifies a time limit in seconds for the command specified by the **–C** flag. If the command does not complete within *RearmCmdTimeout* seconds, it will be sent a SIGTERM signal. If the command does not terminate after an additional 5 seconds, the SIGTERM signal will be followed by a SIGKILL signal. If the **–x** flag is not specified, the command runs until completion.

–U *UserName*
Specifies the user to run the commands specified by the **–c** and **–C** flags. If the **–U** flag is omitted, the commands will run as the user who issued the subscribe request.

–**m** *UserLabel*
Specifies a tag to associate with the subscription. The Problem Management subsystem does not use this tag for anything, so the user can specify anything as a tag. The tag can be retrieved with the **pmanquery** command.

Operands

None.

Description

The Problem Management subsystem allows you to specify that an action takes place as the result of a specific event occurrence within the SP system. The Problem Management subsystem registers for the event with the Event Management subsystem. When the Event Management subsystem reports the occurrence of the event back to Problem Management, the Problem Management subsystem performs the requested action on your behalf. The actions that are performed as the result of an event can be any or all of the following:

- A command is executed
- An entry is written to the AIX error log and BSD syslog
- An SNMP trap is generated

The **pmandef** command is the user interface for making such requests to the Problem Management subsystem. For example, running the following command on node 5:

```
pmandef -s Program_Monitor
-e 'IBM.PSSP.Prog.pcount:NodeNum=12;ProgName=mycmd;UserName=bob:X@0==0'
-r "X@0==1" -c "echo program has stopped >/tmp/myevent.out"
-C "echo program has restarted >/tmp/myrearm.out"
```

causes the command `echo program has stopped >/tmp/myevent.out` to run on node 5 whenever the number of processes named “mycmd” and owned by user “bob” on node 12 becomes 0. When this number increases back to 1, the command `echo program has restarted >/tmp/myrearm.out` runs on node 5.

If you do not want the command to run on the same node from which the **pmandef** command was issued, then use one of the –**h**, –**N**, or –**n** flags.

The following example causes the commands to run on both k21n01 and k21n02 whenever bob's program dies or gets restarted on either nodes 12 or 13.

```
pmandef -s Program_Monitor
-e 'IBM.PSSP.Prog.pcount:NodeNum=12-13;ProgName=mycmd;UserName=bob:X@0==0'
-r "X@0==1" -c /usr/local/bin/start_recovery
-C /usr/local/bin/stop_recovery -h k21n01,k21n02
```

The following example causes the commands to run on nodes 1, 2, 3, and 7 whenever bob's program dies or gets restarted on any of nodes 1, 2, 3, 4, 5, or 13. If bob's program dies on node 4, the command **/usr/local/bin/start_recovery** runs on nodes 1, 2, 3, and 7.

```
pmandef -s Program_Monitor
-e 'IBM.PSSP.Prog.pcount:NodeNum=1-5,13;ProgName=mycmd;UserName=bob:X@0==0'
-r "X@0==1" -c /usr/local/bin/start_recovery
-C /usr/local/bin/stop_recovery -n 1-3,7
```

If you want to define a subscription for more than one node but you want the command to run only on the same node where the event occurs, use the **–h local** option. Consider the following command:

```
pmandef -s Filesystem_Monitor
-e 'IBM.PSSP.aixos.FS.%totused:NodeNum=10-14;VG=myvg;LV=mylv:X>95'
-l "filesystem is almost full" -h local
```

Whenever the file system associated with the "mylv" logical volume and "myvg" volume group on node 11 becomes more than 95 percent full, the text `filesystem is almost full` gets written to the AIX error log and BSD syslog only on node 11. Whenever the same thing occurs on node 12, the same text gets written to the AIX error log and BSD syslog only on node 12. The file system filling up on node 11 is really a separate event than the file system filling up on node 12 or node 13, and the **–h local** option is just a convenient way to define actions for a whole a bunch of events at the same time.

Issuing the **pmandef** command with the **–s** flag to associate an action with an event creates a Problem Management "subscription." When you issue the **pmandef** command to create a Problem Management subscription, the definition of the subscription gets stored in the System Data Repository (SDR) so the definition becomes permanent. As soon as the subscription gets stored in the SDR, the **pmandef** command also requests the affected Problem Management daemons within the SP system to start acting on the new subscription. Since it is possible for some of the nodes affected by this to be down, the **pmandef** command is considered successful once the subscription is stored in the SDR. The failure to reach all of the affected Problem Management daemons is not considered to be an error, because they will eventually pick up the new subscription once they get restarted.

If the Event Management resource variable, instance vector, predicate, or re-arm predicate contains an error, the error will not get discovered until the Problem Management daemon registers for the event with the Event Management subsystem. When this happens, the subscription definition does not automatically get removed from the SDR. You must remove the subscription by using the **pmandef** command with the **–u** flag. The argument to the **–u** flag is the same name that was previously specified as the argument to the **–s** flag.

The **pmandef** command with the **–u** flag removes the subscription definition from the SDR and tells the appropriate Problem Management daemons to stop monitoring for the event associated with that subscription. The **pmandef** command with the **–d** can be used to turn off or "deactivate" a subscription. This also tells the appropriate Problem Management daemons to stop monitoring for the event associated with that subscription, but it does not remove the subscription definition from the SDR. The subscription remains deactivated until you call **pmandef** with the **–a** flag to "activate" the subscription. You can also use the **pmandef** command with the **–q** flag to query the appropriate Problem Management daemons for status about a subscription. This just tells you whether each daemon is currently monitoring for the event associated with that subscription.

The **pmandef** command is based on sysctl, which uses Kerberos for user authentication. As a result, all users of the **pmandef** command must have valid Kerberos credentials. In addition, the user's Kerberos principal must be listed in the **/etc/sysctl.pman.acl** file on the local node, in order to store the subscription in the SDR. It must also be listed on all the nodes that are affected by the new

subscription, in order for the affected Problem Management daemons to be notified of the new subscription. If the user's Kerberos principal is listed only in the **/etc/sysctl.pman.acl** file on the local node, the subscription will be stored in the SDR, but the Problem Management daemons will not act on the new subscription until the next time they are restarted.

The user's Kerberos principal is more than just a mechanism to validate access to the Problem Management subsystem. The Kerberos principal that was in effect when the subscription was created, via the **pmandef -s** command, is stored as part of the subscription definition, and it is used to establish the ownership of that subscription. Modifications to the subscription, via the **pmandef** command with the –**u**, –**d**, and –**a** flags, are only allowed by a user with the same Kerberos principal that is stored in the subscription definition. The Kerberos principal that is stored in the subscription definition is also used by the Problem Management daemon to decide whether the action that is to result from the occurrence of an event should be allowed.

After the Problem Management daemon receives notification that an event has occurred, and before it performs the action for that event, the Problem Management daemon checks to see whether the Kerberos principal that is stored in the subscription definition is authorized to perform the requested action on the node where the Problem Management daemon is running. If the requested action is an entry in the AIX error log and BSD syslog or the generation of an SNMP trap, the Kerberos principal that owns the subscription must be listed in the root user's **$HOME/.klogin** file. If the requested action is the execution of a command, the Kerberos principal must be listed in the **$HOME/.klogin** file of the user that will be used to run the command. The user that will be used to run the command is by default the same user who issued the **pmandef -s** command to create the subscription. A different user can be specified to **pmandef** by using the –**U** flag.

Files

/etc/sysctl.pman.acl
: The Access Control List (ACL) file that authorizes listed Kerberos principals to use Problem Management.

$HOME/.klogin
: The ACL file that authorizes listed Kerberos principals to use a local user account.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference

"Using the Problem Management Subsystem" in *IBM Parallel System Support Programs for AIX: Administration Guide*

Location

/usr/lpp/ssp/bin/pmandef

Related Information

Commands: **pmanquery**, **sysctl**

Examples

In this example, the user mustard is authenticated as the Kerberos principal hotdog@XYZ.COM. If mustard issues the command:

```
pmandef -s Program_Monitor
-e 'IBM.PSSP.Prog.pcount:NodeNum=12;ProgName=mycmd;UserName=relish:X@0==0'
-r "X@0==1" -c /usr/local/bin/start_recovery
-C /usr/local/bin/stop_recovery -n 5 -U ketchup
```

a subscription named “Program_Monitor” and owned by Kerberos principal hotdog@XYZ.COM will be created, if the Kerberos principal hotdog@XYZ.COM is listed in the **/etc/sysctl.pman.acl file** on the local node. The requested actions for this subscription are to occur on node 5, so the Kerberos principal must also be listed in the **/etc/sysctl.pman.acl** file on node 5, in order for the Problem Management daemon on node 5 to act on this subscription immediately. Otherwise, the Problem Management daemon will not act on this subscription until the next time it gets restarted.

This subscription requests that whenever the number of processes named “mycmd” and owned by user relish on node 12 becomes 0, the command start_recovery will run on node 5. When the number of processes increases back to 1, the command stop_recovery will run on node 5. The subscription also requests that the commands start_recovery and stop_recovery are to run on node 5 as the user ketchup. The start_recovery and stop_recovery commands will not run on node 5 unless the Kerberos principal hotdog@XYZ.COM is listed in the **$HOME/.klogin** file of the user ketchup. If the **–U** flag had been omitted, this subscription would have requested the commands to run as the user mustard, and this would have required mustard's **$HOME/.klogin** file to list hotdog@XYZ.COM.

Once the subscription is created, any user who is authenticated as the Kerberos principal hotdog@XYZ.COM can deactivate, activate, or remove this subscription.

To deactivate:

```
pmandef -d Program_Monitor
```

To activate:

```
pmandef -a Program_Monitor
```

To remove:

```
pmandef -u Program_Monitor
```

pmanquery

Purpose

pmanquery – Queries the System Data Repository (SDR) for a description of a Problem Management subscription.

Syntax

pmanquery –**n** {*HandleName* | **all**} [–**k** {*KerberosPrincipal* | **all**}]
[–**q** | –**a** | –**d** | –**t**]

Flags

Flags that specify the scope of the search follow:

–**n** **{***HandleName* | **all}**
Searches for Problem Management subscriptions with the specified handle name that was previously given as the argument to the –**s** flag of the **pmandef** command. The **all** keyword allows any handle name to be selected by the search of the SDR. The **all** keyword cannot be used in conjunction with the –**a**, –**d**, or –**t** flags.

–**k** **{***KerberosPrincipal* | **all}**
Searches for Problem Management subscriptions owned by the specified Kerberos principal. If the –**k** flag is omitted, the caller's Kerberos principal is used. The **all** keyword allows any Kerberos principal to be selected by the search of the SDR. The **all** keyword cannot be used in conjunction with the –**a**, –**d**, or –**t** flags.

Flags that format the output follow:

–**q** Suppresses the output of the header line.

–**a** Lists output by node numbers in column format that represent the set of nodes that have not “deactivated” the subscription identified by the –**n** flag. The **all** keyword cannot be used with either the –**n** or –**k** flag when –**a** is used.

–**d** Lists output by node numbers in column format that represent the set of nodes that have “deactivated” the subscription identified by the –**n** flag. The **all** keyword cannot be used with either the –**n** or –**k** flag when –**d** is used.

–**t** Lists output by node numbers in column format that represent the set of all nodes that recognize the subscription identified by the –**n** flag. This is the sum of the node lists that get printed by the –**a** and –**d** flags. The **all** keyword cannot be used with either the –**n** or –**k** flag when –**t** is used.

Operands

None.

Description

After a Problem Management subscription definition is stored in the SDR by the **pmandef** command, you can use the **pmanquery** command to retrieve the subscription definition. The **pmanquery** command prints the details of the subscription definition in raw format which can then be parsed by other applications.

The –**n** and –**k** flags control the scope of the search for subscriptions in the SDR. Refer to the Examples section of this command for **pmanquery** flag usage information.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

The "Using the Problem Management Subsystem" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*

Location

/usr/lpp/ssp/bin/pmanquery

Related Information

Commands: **pmandef**

Examples

1. To search for a subscription named my_handle that is owned by the caller's Kerberos principal, enter:

   ```
   pmanquery -n my_handle
   ```

2. To search for a subscription named my_handle that is owned by the Kerberos principal hotdog@XYZ.COM, enter:

   ```
   pmanquery -n my_handle -k hotdog@XYZ.COM
   ```

3. To search for all subscriptions that are owned by the Kerberos principal hotdog@XYZ.COM, enter:

   ```
   pmanquery -n all -k hotdog@XYZ.COM
   ```

4. To search for all subscriptions that are owned by the caller's Kerberos principal, enter:

   ```
   pmanquery -n all
   ```

5. To search for all subscriptions named my_handle, enter:

   ```
   pmanquery -n my_handle -k all
   ```

6. To search for all subscriptions, enter:

   ```
   pmanquery -n all -k all
   ```

pmanrmdloadSDR

Purpose

pmanrmdloadSDR – Reads a **pmanrmd** configuration file and loads the information into the System Data Repository (SDR).

Syntax

pmanrmdloadSDR *ConfigFileName*

Flags

None.

Operands

ConfigFileName Specifies a **pmanrmd** configuration file.

Description

The Problem Management subsystem provides 16 resource variables, named IBM.PSSP.pm.User_state1 through IBM.PSSP.pm.User_state16. These are predefined resource variables that have been set aside for users to create their own resource monitors. A resource monitor that you create through Problem Management is a command that gets executed repeatedly by the **pmanrmd** daemon at a specific interval. The standard output from the command is supplied to the Event Management subsystem as the value for the resource variable. You can then use the **pmandef** command to subscribe to events for that resource variable.

The resource variable name, resource monitor command, sampling interval, and list of nodes for which the resource monitor is defined are stored in the SDR. The **pmanrmdloadSDR** command is used to store those definitions in the SDR.

You define your resource monitor to the **pmanrmd** daemon by doing the following:

- Making a copy of the **/spdata/sys1/pman/pmanrmd.conf** sample configuration file.
- Editing your copy of the configuration file. Provide the following:
  - The name of the resource variable (for example, IBM.PSSP.pm.User_state1)
  - The resource monitor command
  - A sampling interval (in seconds)
  - The nodes on which to run the resource monitor command
- Loading the configuration information into the SDR by using the **pmanrmdloadSDR** command.
- Stopping and restarting the **pmanrmd** daemon on the nodes that are affected by this change:

  Control workstation:

  ```
  stopsrc -s pmanrmd.syspar_name
  startsrc -s pmanrmd.syspar_name
  ```

where *syspar_name* is the name of the system partition.

Node:

```
stopsrc -s pmanrmd
startsrc -s pmanrmd
```

For a more complete description of Problem Management resource monitors, refer to the "Using the Problem Management Subsystem" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Files

/spdata/sys1/pman/pmanrmd.conf
A sample **pmanrmd** configuration file.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference

The "Using the Problem Management Subsystem" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*

Location

/usr/lpp/ssp/bin/pmanrmdloadSDR

Related Information

Commands: **pmandef**

pmv

Purpose

pmv – Specifies a parallel file move.

Syntax

pmv [–**w** – | *noderange* | '*hostlist args*'] *mv_args*

Parameters

The **pmv** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

mv_args Specifies arguments to the AIX **rm** or **mv** commands.

Note: The –**i** is not supported (the **dsh** command does not support standard input to remote hosts).

Description

The **pmv** command issues the AIX **mv** command on multiple hosts. The output is formatted so that duplicate output is displayed only once. The **pmv** command uses **dsh** to execute the **mv** command on multiple hosts. The output of the **ls** commands is written to standard output and formatted. The **pmv** command is identical to **pexec mv**.

Since **pmv** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **mv**, **pexec**

Examples

To move a file from each host1, host2, and host3 to a different directory, enter:

```
pmv -w host1,host2,host3 /tmp/shnozzola /etc/shnozzola
```

ppred

Purpose

ppred – Performs a command on those hosts for which a test is satisfied.

Syntax

ppred [–**w** – | *noderange* | '*hostlist args*'] '*ksh test*'
'*true_command*' ['*false_command*']

Parameters

The **ppred** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

'*ksh test***'** **ppred** expects the second argument to be a quoted string in proper syntax to be evaluated via the **ksh test** command. This test is passed to the remote hosts and evaluated on them.

'*true_command***'**
ppred expects the third argument to be a quoted string containing a command to be executed on the hosts for which the test is true.

'*false_command***'**
ppred expects the fourth argument to be a quoted string containing a command to be executed on the hosts for which the test is false. This argument is optional.

Description

The **ppred** command performs a test on remote hosts in parallel. On each host where the test succeeds, a command is run. Optionally, a command can be specified that runs if the test fails.

Since **ppred** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Related Information

Commands: **dsh**, **hostlist**, **test**

Examples

To verify that a file exists and is a regular file on the host occupying the first slot in each of 4 frames, enter:

```
ppred '-s 1-4:1' '-f /etc/passwd' 'echo \'host_name\''
```

pps

Purpose

pps – Specifies a parallel ps command.

Syntax

pps [–**w** – | *noderange* | '*hostlist args*'] *ps_args*

Parameters

The **pps** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

ps_args Specifies arguments to the AIX **ps** command.

Description

The **pps** command uses **dsh** to execute the **ps** command on multiple hosts. The output of the **ls** commands is written to standard output and formatted so that distinct output is presented only once. The **pps** command is identical to **pexec ps**.

Since **pps** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **pexec**, **ps**

Examples

To list processes on each host1, host2, and host3 (described previously), enter:

```
pps -w host1,host2,host3 -ef
```

preparevsd

Purpose

preparevsd – Makes an IBM Virtual Shared Disk available.

Syntax

preparevsd –**a** | *vsd_name*...

Flags

–**a** Specifies that all the IBM Virtual Shared Disks in the stopped state are to be prepared.

Operands

vsd_name Specifies an IBM Virtual Shared Disk. If the IBM Virtual Shared Disk is not in the stopped state, you'll get a failed error message.

Description

The **preparevsd** command brings the specified IBM Virtual Shared Disks from the stopped state to the suspended state. The IBM Virtual Shared Disks are made available. Open and close requests are honored, while read and write requests are held until the IBM Virtual Shared Disks are brought to the active state. If they are in the suspended state, this command leaves them in the suspended state.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Prepare an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/preparevsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **resumevsd**, **startvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Examples

To bring the IBM Virtual Shared Disk **vsd1vg1n1** from the stopped state to the suspended state, enter:

```
preparevsd vsd1vg1n1
```

prm

Purpose

prm – Specifies a parallel file remove.

Syntax

prm [–**w** – | *noderange* | '*hostlist args*'] *rm_args*

Parameters

The **prm** command requires the first flag or parameter on the command line to be a specification of the hosts on which the command is to be executed.

–**w** – Specifies that host names should be read from standard input. Host names can be in any format accepted by SP **rsh**.

noderange Indicates a specification via "node number." The node number corresponds to the position of a node in a frame and its slots. A node number indicates frame and slot. For example, frame 1 slot 1 would be referred to by 1. Frame 2 slot 1 would be node number 17, while frame 3 slot 2 would be 34. If a node occupies more than one slot, either node number refers to that node. Node numbers can be specified as ranges such as 1–3, which would refer to frame 1 slots 1–3, or 23–29,50,2, which would refer to frame 2 slots 7–13, frame 4 slot 2, and frame 1 slot 2. This option is only valid for an SP system.

'*hostlist args***'**
Specifies flags and arguments to be passed to the **hostlist** command. Hostlist allows several ways of listing hosts based on various criteria. Refer to the **hostlist** command.

Note: To use the working collective file specified by the WCOLL environment variable, you must specify a null string as the first argument. Refer to the **dsh** command for more information about a working collective.

rm_args Specifies arguments to the AIX **rm** command.

Description

The **prm** command issues the AIX **rm** command on multiple hosts. The output is formatted so that distinct output is displayed only once. The **prm** command uses **dsh** to execute the **rm** command on multiple hosts. The output of the **ls** commands is written to standard output and formatted. The **prm** command is identical to **pexec rm**.

Since **prm** uses **dsh**, proper authentication and authorization to issue these commands is necessary.

Files

working collective file
See the **dsh** command.

Related Information

Commands: **dsh**, **rm**, **pexec**

Examples

To remove a file from each host1, host2, and host3 (described previously), enter:

```
prm -w host1,host2,host3 /tmp/shnozzola
```

psyslclr

Purpose

psyslclr – Removes entries from **syslog** log files on a set of nodes.

Syntax

psyslclr [–**a**] [–**d** *pids*] [–**e** *endtime*] [–**f** *facilities*] [–**g** *config*] [–**h**] [–**l** *logs*] [–**n** *nodes*] [–**p** *priority*] [–**r** *resources*] [–**s** *startime*] [–**w** *hosts*] [–**y** *days*]

Flags

–**a**	Trims logs on all nodes in the system partition.
–**d** *pids*	Trims records matching the process IDs list.
–**e** *endtime*	Trims records before endtime (*mmddhhmm*).
–**f** *facilities*	Uses the facilities list to parse the **syslog.conf** file.
–**g** *config*	Uses an alternate **syslog.conf** file.
–**h**	Displays usage information.
–**l** *logs*	Trims the list of log files (the **syslog.conf** file is not parsed). (This is lowercase **l**, as in **l**ist.)
–**n** *nodes*	Trims records matching the nodes.
–**p** *priority*	Uses priority value to parse the **syslog.conf** file.
–**r** *resources*	Trims records from the resource list.
–**s** *startime*	Trims records created after startime (*mmddhhmm*).
–**w** *hosts*	Runs the command on a file or list of host names.
–**y** *days*	Trims records more than *days* old.

Operands

None.

Description

Use this command to delete log entries in **syslogd** generated log files. Options allow for selecting the files and records that are trimmed.

The arguments to options –**d**, –**f**, –**l**, –**n**, –**r**, and –**w** can be a comma-delimited or single-quoted, blank-delimited list of values. If the –**l** flag is used, the command will only trim records from the specified list of log file names. If the –**l** flag is not passed, the command will first parse the **syslog** configuration file (the default is /**etc**/**syslog.conf**) to select files for trimming.

The –**f** and –**p** flags can be used to control selecting files in the configuration file. All files found in the configuration file will be trimmed if the –**f** and –**p** flags are not used.

The –**d**, –**e**, –**n**, –**r**, –**s**, and –**y** flags are used to match log entries to be deleted. A record must match a value from each of the flags that are used to be trimmed. If a

flag is not passed, all records match for that field. To delete all records, use the –**y** flag with 0 as the argument. If the –**w** flag begins with a slash (/), it is interpreted as a file containing a list of nodes to execute the command on; otherwise, it can be a list as described previously. If neither the –**a** nor the –**w** flags are used, the command defaults to the local node.

Note: The **syslogd** daemon does not log the year in records timestamps. The comparisons for start and end times are done on a per record basis and could cause unexpected results if the log file is allowed to span more than one year. The **syslogd** daemon is stopped during this process so trimming activity should be planned accordingly. It is then restarted using the default or alternate **syslog** configuration file.

Files

/etc/syslog.conf
syslog daemon configuration file.

/etc/logmgt.acl
Access Control List (ACL) file for **psyslclr** permissions.

Security

You must have a Kerberos principal defined in the **/etc/logmgt.acl** file to run this command.

Related Information

Command: **psyslrpt**

Daemon: **syslogd**

Examples

1. To remove all entries older than 30 days from all **syslog** log files on all nodes in the local system partition, enter:

   ```
   psyslclr -a -y 30
   ```

2. To remove all entries between April 11th and July 23rd that were logged by ftp or snmpd on node k47n10, enter:

   ```
   psyslclr -w k47n10 -s 04110000 -e 07230000 -r ftp,snmpd
   ```

3. To remove all entries from files that may be written by user or mail facilities at a priority level of error or higher on the nodes in the **/tmp/nodelist** file, enter:

   ```
   psyslclr -w /tmp/nodelist -f mail,user -p error -y 0
   ```

psyslrpt

Purpose

psyslrpt – Generates reports of records in **syslog** log files on a set of nodes.

Syntax

psyslrpt [–**a**] [–**d** *pids*] [–**e** *endtime*] [–**f** *facilities*] [–**g** *config*]
[–**h** [–**l** *logs*] [–**n** *nodes*] [–**p** *priority*] [–**r** *resources*]
[–**s** *startime*] [–**w** *hosts*]

Flags

–**a**	Generates the report on all nodes in the system partition.
–**d** *pids*	Reports on records matching the process IDs list.
–**e** *endtime*	Reports on records before *endtime* (*mmddhhmm*).
–**f** *facilities*	Uses the facilities list to parse the **syslog.conf** file.
–**g** *config*	Specifies the use of an alternate **syslog.conf** file.
–**h**	Displays usage information.
–**l** *logs*	Reports on the list of log files (the **syslog.conf** file is not parsed). (This is lowercase **l**, as in **l**ist.)
–**n** *nodes*	Reports records matching the nodes.
–**p** *priority*	Uses priority value to parse the **syslog.conf** file.
–**r** *resources*	Reports records from the resource list.
–**s** *startime*	Reports records created after *startime* (*mmddhhmm*).
–**w** *hosts*	Runs the command on the file or list of host names.

Operands

None.

Description

Use this command to generate reports of log entries in **syslogd** generated log files. Options allow for selecting the files and records that are reported. The arguments to options –**d**, –**f**, –**l**, –**n**, –**r**, and –**w** can be a comma-delimited or single-quoted, blank-delimited list of values. If the –**l** flag is used, the command will report records from the specified list of log file names. If the –**l** flag is not passed, the command will first parse the **syslog** configuration file (the default is /**etc**/**syslog.conf**) to select files for reporting.

The –**f** and –**p** options can be used to control the selecting of files in the configuration file. All files found in the configuration file are reported on if the –**f** and –**p** flags are not used.

The –**d**, –**e**, –**n**, –**r**, and –**s** options are used to match log entries to be reported. A record must match a value from each of these flags that are used to be reported. If a flag is not passed, all records match for that field. If the –**w** argument begins with slash (/), it is interpreted as a file containing a list of nodes to execute the

command on; otherwise, it can be a list as described previously. If neither the –**a** nor –**w** flags are used, the command defaults to the local node.

Files

/etc/syslog.conf
syslog daemon configuration file.

Security

You must be Kerberos authenticated to run this command.

Related Information

Command: **psyslclr**

Daemon: **syslogd**

The *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

1. To report all entries from all **syslog** log files on all nodes in the local system partition starting on March 3rd, enter:

   ```
   psyslrpt -a -s 03030000
   ```

2. To report all entries between April 11th and July 23rd that were logged by ftp or snmpd on node k47n10, enter:

   ```
   psyslrpt -w k47n10 -s 04110000 -e 07230000 -r ftp,snmp
   ```

3. To report entries from the specific log file **/var/adm/SPlogs/SPdaemon.log** with process IDs 10479 or 1157 on nodes k47n12 and k47n15, enter:

   ```
   psyslrpt -w k47n12,k47n15 -d'10479 1157' -l /var/adm/SPlogs/SPdaemon.log
   ```

Note: **syslogd** does not log the year in record timestamps. The comparisons for start and end times are done on a per record basis and could cause unexpected results if the log file is allowed to span more than one year.

rcmdtgt

Purpose

rcmdtgt – Obtains and caches a ticket-granting-ticket for the local realm, with a maximum allowed lifetime, using the service key for the instance of rcmd on the local host.

Syntax

rcmdtgt

Flags

None.

Operands

None.

Description

Use this command to obtain a ticket-granting-ticket with a maximum allowed lifetime, using the service key for **rcmd.***localhost* found in the service key file at /**etc**/**krb-srvtab**. When using SP authentication services, these tickets have an unlimited lifetime. When using AFS authentication services, a maximum of 30 days is enforced.

This command must be run as root and is intended for use in shell scripts and other batch-type facilities. The **rcmdtgt** command retrieves your initial ticket and puts it in the ticket file specified by your KRBTKFILE environment variable. If the KRBTKFILE variable is undefined, your ticket is stored in the /**tmp**/**tkt***uid* file, where *uid* specifies your user identification number.

Note: These tickets are shared by all processes running under the user's IDs. The KRBTKFILE environment variable can be set to change the location of the ticket cache file.

Because the ticket obtained using this command does not expire, the user should be careful to delete the temporary ticket file.

Files

/**etc**/**krb.conf**	Contains the name of the local realm.
/**etc**/**krb-srvtab**	Specifies the service key file.
/**tmp**/**tkt***uid*	Specifies the ticket cache file.

Related Information

Commands: **kdestroy**, **kinit**,

File: **krb.conf**

Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Examples

The following example, excerpted from the sample **script.cust** file, shows how **rcmdtgt** can be used in a shell script to perform the authentication required to use the SP **rcp** command:

```
# set the host name from which you will copy the file.
SERVER='cat /etc/ssp/server_host_name | cut -d" " -f1'

# Define a temporary ticket cache file, then get a ticket
export KRBTKFILE=/tmp/tkt.$$
/usr/lpp/ssp/rcmd/bin/rcmdtgt
#
# Perform kerberos-authenticated rcp
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/etc/resolv.conf /etc/resolv.conf

# Remove the ticket cache file
/bin/rm $KRBTKFILE
```

rcp

Purpose

/usr/lpp/ssp/rcmd/bin/rcp – Transfers files between a local and a remote host.

Note: The remote-to-remote copy is restricted at this time.

Syntax

rcp [–**p**] [–**k** *Kerberos_realm*] {*User*@*Host*:*File* | *Host*:*File* | *File*} {*User*@*Host*:*File* | *Host*:*File* | *File* | *User*@*Host*:*Directory* | *Host*:*Directory* | *Directory*} | [–**r**] {*User*@*Host*:*Directory* | *Host*:*Directory* | *Directory*} {*User*@*Host*:*Directory* | *Host*:*Directory* | *Directory*}

Flags

–**p** Preserves the modification times and modes of the source files in the copies sent to the destination. Without this flag, the **umask** command at the destination modifies the mode of the destination file, and the modification time of the destination file is set to the time the file is received.

When this flag is not used, the **umask** being honored is the value stored in the appropriate database. It is not the value that is set by issuing the **umask** command. The permission and ownership values that result from the **umask** command do not affect those stored in the database.

–**k** *Kerberos_realm*
Specifies that the **rsh** command obtains tickets for the remote host in *realm* instead of the remote host's default realm.

–**r** Recursively copies, for directories only, each file and subdirectory in the source directory into the destination directory.

Operands

Host:*File* Specifies the host name (Host) and file name (File) of the remote destination file, separated by a colon (:).

Note: Because the **rcp** command assumes that a colon (:) terminates a host name, you must insert a backslash (\) before any colons that are embedded in the local file and directory names.

User@*Host*:*File*
Specifies the user name (User@) that the **rcp** command uses to set ownership of the transferred file, the host name (Host), and file name (File) of the remote destination file. The user name entered for the remote host determines the file access privileges the **rcp** command uses at that host.

File Specifies the file name of the local destination file.

Host:*Directory*
Specifies the host name (Host) and directory name (Directory) of the remote destination directory.

Note: Because the **rcp** command assumes that a colon (:) terminates a host name, you must insert a backslash (\) before any colons that are embedded in the local file and directory names.

User@Host:*Directory*
Specifies the user name (User@) that the **rcp** command uses to set ownership of the transferred file, the host name (Host), and directory name (Directory) of the remote destination directory. The user name entered for the remote host determines the file access privileges that the **rcp** command uses at that host.

Directory Specifies the directory name of the local destination directory.

Description

The **rcp** command is used to copy one or more files between the local host and a remote host or between files at the same host.

Remote destination files and directories require a specified Host: parameter. If a remote host name is not specified for either the source or the destination, the **rcp** command is equivalent to the **cp** command. Local file and directory names do not require a Host: parameter.

Note: Because the **rcp** command assumes that a colon (:) terminates a host name, you must insert a backslash (\) before any colons that are embedded in the local file and directory names.

If a Host is not prefixed by a User@ parameter, the local user name is used at the remote host. If a User@ parameter is entered, that name is used.

If the path for a file or directory on a remote host is not specified or is not fully qualified, the path is interpreted as beginning at the home directory for the remote user account. Additionally, any metacharacters that must be interpreted at a remote host must be quoted using a backslash (\), a double quotation mark ("), or a single quotation mark (').

If the **/usr/kerberos/bin/rcp** path does not exist, a link is created from **/usr/lpp/ssp/rcmd/bin/rcp** to **/usr/kerberos/bin/rcp**. The SP **rcp** command uses this path when initially starting an rcp process on a remote host for compatibility purposes.

File Permissions and Ownership

By default, the permissions mode and ownership of an existing destination file are preserved. Normally, if a destination file does not exist, the permissions mode of the destination file is equal to the permissions mode of the source file as modified by the **umask** command (a special command in the Korn shell) at the destination host. If the **rcp** command –**p** flag is set, the modification time and mode of source files are preserved at the destination host.

The user name entered for the remote host determines the file access privileges the **rcp** command uses at that host. Additionally, the user name given to a destination host determines the ownership and access modes of the resulting destination file or files. The remote host allows access if one of the following conditions is satisfied:

- The local user ID is listed as a principal in the authentication database and has performed a **kinit** to obtain an authentication ticket.
- If a **$HOME/.klogin** file exists, it must be located in the user's /**home** directory on the target system. The local user must be listed as well as any users allowed to **rsh** into this account. This file performs a similar function to a local **.rhosts** file. Each line in this file should contain a principal in the form of "principal.instance@realm." If the originating user is authenticated as one of the principals named in **.klogin**, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file.

For security reasons, any **$HOME/.klogin** file must be owned by either the remote user or root, and only the owner should have read and write access.

Should user authentication fail or should the authentication subroutine call fail, the SP **rcp** command passes the arguments to the AIX **rcp** command. In this case, the user needs normal AIX **rcp** access to a host for the command to be successful or the permission denied message is returned. The SP **rcp** command issues several authentication error messages before passing arguments to the AIX **rcp** command. The messages are also visible from system management applications that use the SP **rcp** command.

This command does not support encryption by users.

Files

$HOME/.klogin
Specifies remote users that can use a local user account.

/usr/lpp/ssp/rcmd/bin/rcp
Command executable file.

Prerequisite Information

- Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.
- Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for an overview.
- Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Related Information

SP Commands: **kinit**, **kshd**, **rsh**

AIX Commands: **cp**, **ftp**, **rlogin**, **rsh**, **rshd**, **tftp**, **umask**

Examples

In the following examples, the local user is listed in the authentication database and a ticket was obtained by issuing a **kinit** and the password. In the following examples, host1 is the local host and host2 is the remote host:

1. To copy a local file to a remote host, enter:

```
/usr/lpp/ssp/rcmd/bin/rcp localfile host2:/home/eng/jane
```

The file **localfile** from the local host is copied to the remote host host2.

2. To copy a remote file from one remote host to the local host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rcp host2:/home/eng/jane/newplan/home/eng/mary
   ```

   The file **/home/eng/jane/newplan** is copied from remote host host1 to local host host2.

3. To send the directory subtree from the local host to a remote host and preserve the modification times and modes, enter:

   ```
   /usr/lpp.ssp/rcmd/bin/rcp -p -r report jane@host2:report
   ```

   The directory subtree report is copied from the local host to the home directory of user jane at remote host host2 and all modes and modification times are preserved. The remote file **/home/jane/.rhosts** includes an entry specifying the local host and user name.

4. This example shows how the root user can issue an **rcp** on a remote host. The root user must be in the authentication database and must have already issued **kinit** on the local host. The command is issued at the local host to copy the file, stuff, from node r05n07 to node r05n05.

   ```
   /usr/lpp/ssp/rcmd/bin/rsh r05n07 'export KRBTKTFILE=/tmp/rcmdtkt$$; \
   /usr/lpp/ssp/rcmd/bin/rcmdtgt; \
   /usr/lpp/ssp/rcmd/bin/rcp /tmp/stuff r05n05:/tmp/stuff;'
   ```

   The root user sets the KRBTKTFILE environment variable to the name of a temporary ticket-cache file and then obtains a service ticket by issuing the **rcmdtgt** command. The **rcp** command uses the service ticket to authenticate from host r05n07 to host r05n05.

removehsd

Purpose

removehsd – Removes one or more data striping devices, the IBM Virtual Shared Disks associated with them, and the System Data Repository (SDR) information for IBM Virtual Shared Disks on the associated nodes. If the only volume groups defined on the associated nodes are the ones used by the data striping devices that are being removed, those volume groups are also removed.

Syntax

removehsd {–**v** {*hsd_names* | –**a**}} [–**f**]

Flags

–**v** Specifies the data striping device (HSD) name or names that are to be removed by this command.

–**a** Specifies that the command should remove all HSDs in the system or system partition.

–**f** Forces the system to unconfigure the HSD and its underlying IBM Virtual Shared Disks and remove them. If –**f** is not specified and any of the IBM Virtual Shared Disks that constitute the HSD to be removed are configured or the HSD itself is configured, the command fails.

Operands

None.

Description

Use this command to remove the logical volumes associated with IBM Virtual Shared Disks in the set of HSDs. The order in which the IBM Virtual Shared Disks that make up the HSDs and the HSDs themselves are removed is the reverse of the order in which they were created.

If the IBM Virtual Shared Disk or HSD is configured on any of the nodes on the system partition, this command fails, unless the –**f** flag is specified.

Files

/usr/lpp/csd/bin/removehsd
Specifies the file that contains the command.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **createhsd**, **removevsd**

Examples

To unconfigure and remove the IBM Virtual Shared Disks associated with the HSD DATA and remove the HSD as well, type:

```
removehsd -d DATA -f
```

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Remove a Hashed Shared Disk option.

removevsd

Purpose

removevsd – Removes a set of IBM Virtual Shared Disks that are not part of any data striping device (HSD).

Syntax

removevsd –**v** *vsd_names* | –**a** [–**f**]

Flags

–**v** Specifies the IBM Virtual Shared Disk name or names that are to be removed by this command.

–**a** Specifies that the command should remove all IBM Virtual Shared Disks in the system or system partition.

–**f** Forces the system to unconfigure the IBM Virtual Shared Disks and remove them. If –**f** is not specified and any of the IBM Virtual Shared Disks that are to be removed are configured, the command fails.

Operands

None.

Description

Use this command to remove the logical volumes associated with the IBM Virtual Shared Disks and update the backup nodes' Object Data Managers (ODMs), if any exist. The IBM Virtual Shared Disk information will be deleted from the System Data Repository (SDR). The removal of the IBM Virtual Shared Disks is done in the reverse of the order in which they were created. Volume groups are not removed with this command.

If the IBM Virtual Shared Disk is configured on any of the nodes on the system partition, this command fails, unless the –**f** flag is specified.

Note: This command fails if one of the IBM Virtual Shared Disks named in *vsd_names* belongs to an HSD. To remove IBM Virtual Shared Disks that belong to an HSD, use **removehsd**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Remove an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/removevsd
Specifies the file that contains the command.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **createvsd**, **removehsd**

Examples

To unconfigure and remove all defined IBM Virtual Shared Disks in a system or system partition, enter:

```
removevsd -a -f
```

resumevsd

Purpose

resumevsd – Activates an available IBM Virtual Shared Disk.

Syntax

resumevsd [–**p** | –**b**] –**a** | *vsd_name* ...

Flags

–**p** Specifies that the primary server node defined for the global volume group is to be the active server.

–**b** Specifies that the secondary server node defined for the global volume group is to be the active server.

Note: This flag is used only by the IBM Recoverable Virtual Shared Disk product.

–**a** Specifies that all the IBM Virtual Shared Disks that have been defined are to be resumed.

Operands

vsd_name Specifies an IBM Virtual Shared Disk.

Description

The **resumevsd** command brings the specified IBM Virtual Shared Disks from the suspended state to the active state. The IBM Virtual Shared Disks remains available. Read and write requests which had been held while the IBM Virtual Shared Disk was in the suspended state are resumed.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Resume an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/resumevsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

1. If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

   See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

2. The –**b** flag is used only by the IBM Recoverable Virtual Shared Disk product.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **startvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Examples

To bring the IBM Virtual Shared Disk **vsd1vg1n1** from the suspended state to the active state, enter:

```
resumevsd vsd1vg1n1
```

rsh

Purpose

/usr/lpp/ssp/rcmd/bin/rsh – Executes the specified command at the remote host.

Syntax

{**rsh** | **remsh**} *RemoteHost* [–**n**] [–**l** *user*] [–**k** *Kerberos_realm*] *command*

Flags

–**n** Specifies that the **rsh** command should not read from standard input. Use this flag if you are in C shell and run the **rsh** command in the background.

–**l** *user* Specifies that the **rsh** command should log into the remote host as the named *user* instead of the local user name. If this flag is not specified, the local and remote user names are the same. (This is lowercase **l**, as in **l**ist.)

–**k** *Kerberos_realm*
Specifies that the **rsh** command obtains tickets for the remote host in *realm* instead of the remote host's default realm.

Operands

RemoteHost
Specifies the **rsh** destination host that runs *command*.

command Specifies the name of the command to be executed at the *RemoteHost*.

Description

The **rsh** command executes *command* at *RemoteHost*. If *command* is not specified, the usage message is displayed. The **rsh** command sends standard input from the local command line to the remote command and receives standard output and standard error from the remote command.

Note: Since any input to the remote command must be specified on the local command line, you cannot use the **rsh** command to execute an interactive command on a remote host. If you need to execute an interactive command on a remote host, use the AIX **rlogin** command. If you do not specify a *command*, the **rsh** command displays the usage message and exits.

Remote Command Execution

While the remote command is executing, pressing the INTERRUPT, TERMINATE, or QUIT key sequences sends the corresponding signal to the remote process. However, pressing the STOP key sequence stops only the local process. Normally, when the remote command terminates, the local rsh process terminates.

To have shell metacharacters interpreted on the remote host, place the metacharacters inside double quotes (" "). Otherwise, the metacharacters are interpreted by the local shell.

When using the **rsh** command, you can create a link to a path (that you have permission to write) using a host name as the link name. For example,

```
ln -s /usr/lpp/ssp/rcmd/bin/rsh host_name
```

At the prompt, if you enter the host name and a parameter (command), the **rsh** command is used to remotely execute the command specified on the command line on the remote host called host name. For example, if you are linked to remote host opus and want to perform the date command, you would enter:

```
opus date
```

At the prompt, if you enter the host name with no parameters, **rsh** displays the usage message and exits.

Files

$HOME/.klogin
Specifies remote users that can use a local user account.

/usr/lpp/ssp/rcmd/bin/rsh
Command binary file.

/usr/lpp/ssp/rcmd/bin/remsh
Link to the **/usr/lpp/ssp/rcmd/bin/rsh** executable file.

Security

If you do not specify the **–l** flag, the local user name is used at the remote host. If **–l** *user* is entered, the specified user name is used at the remote host. In either case, the remote host allows access only if at least one of the following conditions is satisfied:

- The local user ID is listed as a principal in the authentication database and had performed a **kinit** to obtain an authentication ticket.
- If a **$HOME/.klogin** file exists, it must be located in the local user's **/home** directory on the target system. The local user must be listed as well as any users or services allowed to rsh into this account. This file performs a similar function to a local **.rhosts** file. Each line in this file should contain a principal in the form of "principal.instance@realm." If the originating user is authenticated as one of the principals named in **.klogin**, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file.

For security reasons, any **$HOME/.klogin** file must be owned by either the remote user or root, and only the owner should have read and write access.

Should user authentication fail or should the authentication subroutine call fail, SP **rsh** passes the arguments to the AIX **rsh** command. In this case, the user needs normal AIX **rsh** access to a host for the command to be successful or the permission denied message is returned. The SP **rsh** command issues several authentication error messages before passing arguments to the AIX **rsh** command. The messages are also visible from system management applications that use the SP **rsh** command.

This command does not support encryption by users.

Prerequisite Information

- Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.
- Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for an overview.
- Refer to Chapter 2, "SP Files and Other Technical Information" section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Related Information

SP Commands: **kinit**, **kshd**, **rcp**

AIX Commands: **rcp**, **rlogin**, **rshd**, **telnet**

Examples

In the following examples, the local user is listed in the authentication database and has obtained a ticket by issuing a **kinit** and the password.

1. To check the amount of free disk space on a remote host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 df
   ```

   The amount of free disk space on host2 is displayed on the local system.

2. To append a remote file to another file on the remote host, place the >> metacharacters in quotation marks, and enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 cat test1 ">>" test2
   ```

   The file test1 is appended to test2 on remote host host2.

3. To append a remote file at the remote host to a local file, omit the quotation marks, and enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 cat test2 >> test3
   ```

   The remote file test2 on host2 is appended to the local file test3.

4. To append a remote file to a local file and use a remote user's permissions at the remote host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 -l jane cat test4 >> test5
   ```

   The remote file test4 is appended to the local file test5 using user jane's permissions at the remote host.

   This example shows how the root user can issue an **rcp** on a remote host. The root user must be in the authentication database and must have already issued **kinit** on the local host. The command is issued at the local host to copy the file, stuff, from node r05n07 to node r05n05.

   ```
   /usr/lpp/ssp/rcmd/bin/rsh r05n07 'export KRBTKTFILE=/tmp/rcmdtkt$$; \
   /usr/lpp/ssp/rcmd/bin/rcmdtgt; \
   /usr/lpp/ssp/rcmd/bin/rcp /tmp/stuff r05n05:/tmp/stuff;'
   ```

   The root user sets the KRBTKTFILE environment variable to the name of a temporary ticket-cache file and then obtains a service ticket by issuing the **rcmdtgt** command. The **rcp** uses the service ticket to authenticate from host r05n07 to host r05n05.

SDR_test

Purpose

SDR_test – Verifies that the installation and configuration of the System Data Repository (SDR) component of the SP system completed successfully.

Syntax

SDR_test [–**q**] [–**l** *log_file*]

Flags

–**q** Specifies quiet mode; suppresses output to standard error.

–**l** *log_file* Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

Operands

None.

Description

This command verifies that the SDR is functioning properly. After clearing out a class name, it creates the class, performs various SDR tasks, and removes the class when done.

A return code of 0 indicates that the test completed as expected; otherwise it returns the number of failures. If you do not specify the –**q** flag, a message is displayed on standard output that indicates the success or failure of the tests. In either case, the command returns 0 if successful, 1 if not. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/SDR_test.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Data Repository option.

Files

/usr/lpp/ssp/bin/SDR_test
Path name of this command.

/var/adm/SPlogs/SDR_test.log
Default log file.

Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SYSMAN_test**, **spmon_ctest**, **spmon_itest**

Examples

To verify the System Data Repository following installation, saving error messages in **sdr_err** in the current working directory, enter:

```
SDR_test -l sdr_err
```

SDRAddSyspar

Purpose

SDRAddSyspar – Creates a new daemon using the System Resource Controller (SRC). The new daemon creates a subdirectory under the **/spdata/sys1/sdr/partitions** directory.

Syntax

SDRAddSyspar *IP_address*

Flags

IP_address Specifies a TCP dotted decimal address (real or alias).

Operands

None.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates a new instance of the SDR daemon and passes it the IP address of the system partition. It does not perform all of the system management tasks involved in creating a system partition.

SDRArchive

Purpose

SDRArchive – Archives the entire contents of the System Data Repository (SDR), except for the archives directory, for later retrieval.

Syntax

SDRArchive [*append_string*]

Flags

None.

Operands

append_string If specified, the *append_string* is appended to the name of the backup file.

Description

Use this command to tar the contents of the SDR and put the file in the **/spdata/sys1/sdr/archives** subdirectory. You might want to mount this directory from another machine or physical disk drive to protect against a failure of the drive that the SDR exists on. The file name is **backup.JULIANdate.***HHMM.append_string*, where **JULIANdate.***HHMM* is a number or string uniquely identifying the date and time of the archive and *append_string* is the argument entered in the command invocation, if specified.

SDRChangeAttrValues

Purpose

SDRChangeAttrValues – Changes attribute values of one or more objects.

Syntax

SDRChangeAttrValues *class_name* [*attr*==*value* ...] *attr*=*value* ...

Flags

None.

Operands

class_name Identifies the target object and checks the class to see if it is a system class or a partitioned class.

attr==*value* Specifies attribute values required for the change to be made (note double equal signs signifying comparison.)

attr=*value* Specifies target attribute and value to be assigned.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command changes one or more attribute values in a specified object with certain other attribute values.

SDRClearLock

Purpose

SDRClearLock – Unlocks a System Data Repository (SDR) class.

Syntax

SDRClearLock *class_name*

Flags

None.

Operands

class_name Identifies the target class and removes the lock on that class if a lock exists.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

Use this command when a process that obtained a lock ends abnormally and fails to unlock the class.

SDRCreateAttrs

Purpose

SDRCreateAttrs – Creates new attributes for a System Data Repository (SDR) class.

Syntax

SDRCreateAttrs *class_name attr*=*datatype* ...

Flags

None.

Operands

class_name Identifies the target object.

attr **=** *datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates one or more new attributes for a target class.

SDRCreateClass

Purpose

SDRCreateClass – Creates a partitioned class.

Syntax

SDRCreateClass *class_name attr*=*datatype* ...

Flags

None.

Operands

class_name Identifies the new object class.

attr **=** *datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates a partitioned class and defines its attributes.

SDRCreateFile

Purpose

SDRCreateFile – Reads the specified AIX file and puts it in the System Data Repository (SDR) under the specified SDR file name.

Syntax

SDRCreateFile *AIX_filename SDR_filename*

Flags

None.

Operands

AIX_filename
Identifies the AIX file name to be written to the SDR.

SDR_filename
Specifies the name of the new SDR file.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates a partitioned SDR file from an AIX file. Use **SDRCreateSystemFile** to create a system file. Use **SDRRetrieveFile** to retrieve the file.

SDRCreateObjects

Purpose

SDRCreateObjects – Creates new objects in a system class or a partitioned class.

Syntax

SDRCreateObjects *class_name attr*=*value* ...

Flags

None.

Operands

class_name Identifies the class of the new objects.

attr=*value* Specifies the target attribute and value to be assigned.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates one or more new objects. Not all attributes for an object need to be specified in this call; however, a subset of the attributes that uniquely identify this object must be entered at this time.

SDRCreateSystemClass

Purpose

SDRCreateSystemClass – Creates a system class.

Syntax

SDRCreateSystemClass *class_name attr*=*datatype* ...

Flags

None.

Operands

class_name Identifies the new object class.

attr **=** *datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command creates a system class and defines its attributes.

SDRCreateSystemFile

Purpose

SDRCreateSystemFile – Creates a file that can be retrieved from any system partition.

Syntax

SDRCreateSystemFile *AIX_filename SDR_filename*

Flags

None.

Operands

AIX_filename
Specifies the AIX file name.

SDR_filename
Specifies the System Data Repository (SDR) file name.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command reads the AIX file and puts it in the repository under the SDR file name. Note that only ASCII files can be saved. Results are unpredictable if binary files are used with this command. Clients connected to any system partition can read this file.

Use **SDRRetrieveFile** to retrieve this file. If a system file and a partitioned file exist with the same name, the partitioned file will be returned from **SDRRetrieveFile**.

SDRDeleteFile

Purpose

SDRDeleteFile – Deletes the specified System Data Repository (SDR) file.

Syntax

SDRDeleteFile *SDR_filename*

Flags

None.

Operands

SDR_filename
Specifies the name of the SDR file to be deleted.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command deletes the partitioned file *SDR_filename*, if it exists. If the *SDR_filename* partitioned file does not exist, it will delete the *SDR_filename* system file. This command will not delete both the partitioned file and the system file.

SDRDeleteObjects

Purpose

SDRDeleteObjects – Deletes objects from the System Data Repository (SDR).

Syntax

SDRDeleteObjects *class_name* [*attr*==*value* ...]

Flags

None.

Operands

class_name Identifies the class of the object to be deleted.

attr==*value* Specifies specific attribute values qualifying the object for deletion.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command deletes one or more objects. All objects in the specified class with attribute values matching those specified are deleted. If no *attr*==*value* pairs are specified, this command will match all objects in the class and all objects will be deleted.

SDRGetObjects

Purpose

SDRGetObjects – Sends contents of attributes in specified object to standard output.

Syntax

SDRGetObjects [–**G**] [–**x**] [–**q**] [–**d** *delimiter*] *class_name* [*attr*==*value* ...] [*attr* ...]

Flags

–**G**	For a partitioned class, returns the objects that match the *attr*==*value* arguments in the class specified from **all** system partitions. For system classes, the –**G** option has no effect.
–**x**	Inhibits the output of the header line.
–**q**	Specifies quiet mode; suppresses message output to standard error.
–**d** *delimiter*	Allows the user to specify a delimiter in the output.

Operands

class_name	Identifies the target class.
attr==*value*	Specifies attribute values qualifying the objects for the operation (note the double equal signs, which signify comparison.)
attr	Specifies which attribute values should be returned by the command. The order of these arguments is the order they are written in the output. If no *attr* arguments are entered, all attributes will be selected.

Description

This command retrieves and sends to standard output attribute values in the specified objects.

Examples

1. To query the SDR Adapter class for the node number and network address of all High Performance Switch adapters, enter:

   ```
   SDRGetObjects -G Adapter adapter_type==css0 node_number netaddr
   ```

   You should receive output similar to the following:

```
node_number netaddr
          1 129.40.102.129
          3 129.40.102.131
          5 129.40.102.133
          6 129.40.102.134
          7 129.40.102.135
          8 129.40.102.136
          9 129.40.102.137
         10 129.40.102.138
         11 129.40.102.139
         12 129.40.102.140
         13 129.40.102.141
         14 129.40.102.142
         15 129.40.102.143
         16 129.40.102.144
```

2. To determine the reliable host name, switch node number, and switch chip port of every Node object, enter:

   ```
   SDRGetObjects -G Node reliable_hostname switch_node_number \
   switch_chip_port
   ```

   You should receive output similar to the following:

   ```
   reliable_hostname switch_node_number switch_chip_port
   k3n01.hpssl.kgn.ibm.com            0            3
   k3n03.hpssl.kgn.ibm.com            2            0
   k3n05.hpssl.kgn.ibm.com            4            1
   k3n06.hpssl.kgn.ibm.com            5            0
   k3n07.hpssl.kgn.ibm.com            6            2
   k3n08.hpssl.kgn.ibm.com            7            3
   k3n09.hpssl.kgn.ibm.com            8            3
   k3n10.hpssl.kgn.ibm.com            9            2
   k3n11.hpssl.kgn.ibm.com           10            0
   k3n12.hpssl.kgn.ibm.com           11            1
   k3n13.hpssl.kgn.ibm.com           12            1
   k3n14.hpssl.kgn.ibm.com           13            0
   k3n15.hpssl.kgn.ibm.com           14            2
   k3n16.hpssl.kgn.ibm.com           15            3
   ```

3. To save each node's node number and hardware Ethernet addresses (which is needed for netboot) in a file called **bootptab.info** without the SDR header class information, enter:

   ```
   SDRGetObjects -G -x Node node_number hdw_enet_addr > /etc/bootptab.info
   ```

   You should receive output similar to the following:

   ```
          1 02608C2D58D2
          3 10005AFA2375
          4 10005AFA22CE
          5 10005AFA22B2
          6 10005AFA2410
          7 10005AFA223F
          8 10005AFA2417
          9 02608C2DA0C7
         11 02608C2D9F62
         13 02608C2D9E75
         15 10005AFA1B03
         16 10005AFA2B9B
   ```

SDRListClasses

Purpose

SDRListClasses – Lists the class names in the System Data Repository (SDR).

Syntax

SDRListClasses

Flags

None.

Operands

None.

Description

This command outputs all of the class names (system and partitioned) currently defined in the SDR to standard output.

SDRListFiles

Purpose

SDRListFiles – Lists all of the files in the system file area first, then lists all of the files in the system partition area.

Syntax

SDRListFiles

Flags

None.

Operands

None.

Description

This command outputs all the system partition files first, then the system files.

SDRMoveObjects

Purpose

SDRMoveObjects – Moves objects from one system partition to another.

Syntax

SDRMoveObjects *source_syspar target_syspar class_name* [*attr*==*value* ...]

Flags

None.

Operands

source_syspar
Specifies the system partition from which objects are to be moved.

target_syspar
Specifies the system partition to which objects should be moved.

class_name Identifies the target object.

attr==*value* Specifies attribute values to be moved.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command moves any objects in *class_name* that match all of the *attr*==*value* pairs from the *source_syspar* to the *target_syspar*.

SDRRemoveSyspar

Purpose

SDRRemoveSyspar – Removes all of the partitioned classes in the System Data Repository (SDR) associated with the system partition whose address is *IP_address*. It removes the daemon that serves this system partition using the System Resource Controller (SRC).

Syntax

SDRRemoveSyspar *IP_address*

Flags

None.

Operands

IP_address Specifies the dotted decimal address (real or alias) of a system partition.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command deletes a system partition in the SDR. It does not perform all of the system management tasks involved in deleting a system partition.

SDRReplaceFile

Purpose

SDRReplaceFile – Replaces the specified System Data Repository (SDR) file with the specified AIX file.

Syntax

SDRReplaceFile *AIX_filename SDR_filename*

Flags

None.

Operands

AIX_filename
Identifies the AIX file name to be written to the SDR.

SDR_filename
Specifies the name of the SDR file to be overwritten.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

This command searches first for a partitioned file, then for a system file, and replaces the first one found.

SDRRestore

Purpose

SDRRestore – Extracts the contents of the archived System Data Repository (SDR).

Syntax

SDRRestore *archive_file*

Flags

None.

Operands

archive_file Indicates the name of the archive file. The file name is **backup.JULIANdate.***HHMM*, where **JULIANdate.***HHMM* is a number or string uniquely identifying the time of the archive.

Description

Warning

Warning: The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause corruption of system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

Use this command to remove the contents of the SDR and retrieve the archived contents of the *archive_file*. The *archive_file* must be in the **/spdata/sys1/sdr/archives** directory. Any new SDR daemons that represent partitions in the restored SDR are then started and any daemons that are not in the new SDR are stopped.

Restrictions

The **SDRRestore** command should be not issued from a PSSP Version 1 Release 2 client if the control workstation is not at PSSP Version 1 Release 2.

In PSSP Version 1 Release 2, the *archive_file* is not brought across from the client to the server. In subsequent versions, the *archive_file* is expected to be in the **/spdata/sys1/sdr/archives** file on the control workstation. If a PSSP Version 1 Release 2 client issues this command, all of the classes are created as partitioned classes, which is not the desired result.

Related Information

Command: **SDRArchive**

SDRRetrieveFile

Purpose

SDRRetrieveFile – Retrieves the specified System Data Repository (SDR) file into an AIX file.

Syntax

SDRRetrieveFile *SDR_filename AIX_filename*

Flags

None.

Operands

SDR_filename
Specifies the name of the SDR file to be retrieved.

AIX_filename
Identifies the name of the AIX file to be written.

Description

This command searches first for a partitioned file, then for a system file if a partitioned file was not found.

SDRWhoHasLock

Purpose

SDRWhoHasLock – Returns transaction ID of lock on specified class.

Syntax

SDRWhoHasLock *class_name*

Flags

None.

Operands

class_name Identifies the target object class.

Description

The lock transaction ID returned from this command takes the form *host_name:pid:session*, where *host_name* is the long name of the machine running the process with the lock, *pid* is the process ID of the process that has the lock, and *session* is the number of the client's session with the System Data Repository (SDR).

seqfile

Purpose

seqfile – Creates node sequence files for system startup and shutdown using information in the System Data Repository (SDR).

Syntax

seqfile [–**b**]

Flags

–**b** Includes lines for boot/install servers as well as /**usr** servers. Specify this option to create lines for /**etc**/**cstartSeq**.

Operands

None.

Description

seqfile uses information in the SDR to determine dependencies of SP nodes on /**usr** and, optionally, and boot/install servers, and to write the dependencies to standard output in the format of the node sequence files /**etc**/**cstartSeq** and /**etc**/**cshutSeq**.

/**usr** servers must shut down after and start up before their clients. Boot-install servers must start up before their clients. The node sequence files, /**etc**/**cshutSeq** and /**etc**/**cstartSeq**, have lines that describe these dependencies. The **seqfile** command eliminates the need for you to create these files from scratch. If the nodes in your system have sequencing dependencies in addition to those related to boot/install and /**usr** servers and clients, you can edit the output of **seqfile** to define those relationships.

seqfile defines only the nodes that have dependencies; if there are no /**usr** or boot/install dependencies, **seqfile** generates no output.

If you do not have a /**etc**/**cstartSeq** or /**etc**/**cshutSeq** file, the **cstartup** and **cshutdown** commands use **seqfile** to determine the default startup or shutdown sequence.

Files

The following files reside on the control workstation:

/**usr**/**lpp**/**ssp**/**bin**/**seqfile**
The **seqfile** command.

/**etc**/**cshutSeq** Describes the sequence in which the nodes should be shut down. Nodes not listed in the file are shut down concurrently with listed nodes. If the file is empty, all nodes are shut down concurrently. If the file does not exist, **cshutdown** uses the output of **seqfile** as a temporary sequencing default.

/etc/cstartSeq Describes the sequence in which the nodes should be started. Nodes not listed in the file are started up concurrently with listed nodes. If the file is empty, all nodes are started up concurrently. If the file does not exist, **cstartup** uses the output of **seqfile** as a temporary sequencing default.

Related Information

Commands: **cshutdown**, **cstartup**

Examples

1. To create the node sequence file for system startup from information in the SDR, enter:

   ```
   seqfile -b > /etc/cstartSeq
   ```

2. To create the node sequence file for system shutdown from information in the SDR, enter:

   ```
   seqfile > /etc/cshutSeq
   ```

3. To view the sequence used during system shutdown in the absence of a **/etc/cshutSeq** file, enter:

   ```
   seqfile | more
   ```

services_config

Purpose

services_config – Configures designated services on nodes or the control workstation.

Syntax

services_config

Flags

None.

Operands

None.

Description

Use this command to configure SP services on the node or control workstation.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/install/bin/services_config

Related Information

Commands: **setup_server**, **spbootins**

Examples

To configure the SP services on a node or the control workstation, enter:

```
services_config
```

sethacws

Purpose

sethacws – Sets the state of the control workstation.

Syntax

sethacws *state*

Flags

None.

Operands

state Specifies a number of the set: 0, 1, 2, 16, 32.

Description

Use this command to set the current state of the control workstation. It is valid only when issued on the control workstation. When the command is executed and the calling process is not on a control workstation, an error occurs.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

Exit Values

0 Indicates successful completion of the command.

1 Indicates that the command could not access the control workstation state location.

2 Indicates that the command was executed with a control workstation state that was not valid.

3 Indicates that the command was not executed on a control workstation.

The following are the valid state values and their defined control workstation state:

0 Indicates that the configuration is not an HACWS configuration, but is a control workstation.

1 Indicates that this is the primary control workstation, but not the active control workstation.

2 Indicates that this is the primary and active control workstation.

16 Indicates that this is the backup control workstation and not the active control workstation.

32 Indicates that this is the backup and active control workstation.

Security

You must have root privilege to run this command.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on the HACWS option.

Location

/usr/bin/sethacws

Related Information

Command: **lshacws**

Subroutines: **hacws_set**, **hacws_stat**

Examples

1. To set the control workstation state as a backup and active control workstation, enter:

   ```
   sethacws 32
   ```

2. To set the control workstation state as a backup and inactive control workstation, enter:

   ```
   sethacws 16
   ```

3. To set the control workstation state as a primary and active control workstation, enter:

   ```
   sethacws 2
   ```

4. To set the control workstation state as a primary and inactive control workstation, enter:

   ```
   sethacws 1
   ```

5. To set the control workstation state as a control workstation but not an HACWS configuration, enter:

   ```
   sethacws 0
   ```

setup_authent

Purpose

setup_authent – Sets up a workstation to use SP authentication services.

Syntax

setup_authent

Flags

None.

Operands

None.

Description

The **setup_authent** command configures SP authentication services during SP installation on the control workstation and on other IBM RS/6000 workstations connected to an SP system. It is not executed on SP nodes, where authenticated client services are automatically installed. Executing this command invokes an interactive dialog, in which instructions for the various steps are displayed and various utility programs are invoked to accomplish the configuration.

There are several ways that **setup_authent** can configure these services. The method chosen is based on runtime choice, the combination of SP options installed on the workstation, and the contents of any predefined authentication configuration file that you have supplied.

Primary Server: When the local system is to be configured as the primary server, both **ssp.clients** and **ssp.authent** SP options must have been installed. You may supply the configuration files, **/etc/krb.conf** and **/etc/krb.realms**, or let the system create one that lists the local system as the sole authentication server in the local realm. This command creates the files used by the authentication and ticket granting services. These include the configuration files, the authentication database files, and the master key cache file. The server daemon, **kerberos**, is added to the inittab and started.

The administration of the authentication database is handled by the **kadmind** daemon, which is also added to inittab and started. The **setup_authent** command requires you to define the initial principal who administers the database. Access control list files are created containing this name, to be used by **kadmind** for authorization.

This command invokes **kinit** to log you in as this administrator, in order to define additional principals used by the SP authenticated services for monitoring and system administration. A server key file is created for use by the monitor commands, SP remote commands, and Sysctl remote command execution facility.

Backup Server: When the local workstation is to be configured as a secondary server, **ssp.clients** and **ssp.authent** must be installed. You must supply the configuration files, listing the local host as a slave server and some other

workstation as the primary authentication server. The primary server must be configured and running and be available by standard TCP/IP connection to the local host.

You are required to authentication your identity as the administrative user that you defined when you configured the primary server. The service principals for the local host are added to the primary database, and the server key file is created for them. Then the **kpropd** daemon is used in conjunction with the **kprop** command (executed remotely on the primary server) to copy the master database onto the local system. The server daemon, **kerberos**, is then added to the inittab and started.

Authentication Server: When the local host is to be configured only to provide authentication client services, just **ssp.clients** needs to be installed. As in the case of the slave server, you must supply the configuration files. In this case, however, the local host is not listed as a server. **setup_authent** simply requires the information to know how to get to the primary authentication server (already configured and accessible).

You are required to authenticate your identity as the administrative user that you defined when you configured the primary server. The service principals for the local host are added to the primary database, and the server key file is created for them.

Using AFS Authentication Services: When AFS authentication is to be configured, the local host must have already been established as either an AFS server or an AFS client. The **CellServDB** and **ThisCell** files are expected to exist in the /**usr**/**vice**/**etc** directory (or linked to that path). **ssp.clients** is the only required SP authentication option. When **setup_authent** finds these AFS configuration files on the local system, it allows you the choice of whether to use AFS authentication. If you choose not to use AFS, processing follows one of the other three variations described previously. When using AFS, you must supply an AFS user name and password that is a valid administrative ID in the local AFS cell. **setup_authent** creates the local service principals in the AFS database and creates a server key file for the SP authenticated services to use on the local host.

If you choose AFS authentication, you must do so for all workstations you configure with **setup_authent**, including the control workstation for your SP system.

You can reexecute **setup_authent** to change the configuration of your authentication services, but you add varying degrees of risk to system operations depending on how far you have progressed in the installation of the control workstation and nodes. Running it again on the control workstation prior to executing **install_cw** is not a problem. Reconfiguring a client workstation has little risk of disruption. A slave can be reconfigured provided the primary server is available. If the primary server must be reconfigured, all slave and client systems have to be reconfigured after the new primary server is up. If the control workstation is an authentication server, you have to recustomize any SP nodes previously booted, after running **setup_authent**.

Files

/.k Master key cache file.

/etc/krb.conf Authentication configuration file.

/etc/krb.realms
Authentication configuration file.

/etc/krb-srvtab Server key file.

/usr/kerberos/admin_acl.{add,get,mod}
Access Control List files.

/var/kerberos/database/principal.pag, /var/kerberos/database/principal.dir
Authentication database files.

/usr/vice/etc/CellServDB, /usr/vice/etc/ThisCell
AFS configuration files.

Related Information

Commands: **add_principal**, **ext_srvtab**, **kadmind**, **kdb_edit**, **kdb_init**, **kdb_util**, **kerberos**, **kinit**, **klist**, **krb_conf**, **krb_realms**, **ksrvutil**, **kstash**

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

setup_CWS

Purpose

setup_CWS – Updates control workstation files and directories for installation tasks.

Syntax

setup_CWS [–**h**]

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken.

Operands

None.

Description

Use this command to update control workstation files and directories for installation tasks. This includes control workstation-specific Kerberos and other files. This command can only be run on the control workstation.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/setup_CWS

Related Information

Commands: **setup_server**

Examples

To update the control workstation environment for installation, enter:

```
setup_CWS
```

setup_logd

Purpose

setup_logd – Sets up the logging daemon (**splogd**). This is called by installation scripts when the IBM RS/6000 control workstation is installed. It can also be run by root on a different workstation to have **splogd** spawned by the System Resource Controller (SRC).

Syntax

setup_logd

Flags

None.

Operands

None.

Description

To run the **splogd** logging daemon on a workstation other than the control workstation, install the **ssp.clients** option on that workstation and run **setup_logd**. You may want to do this in order to do the following:

1. Offload error logging from the control workstation
2. Have your own script called when a state change on a particular variable or variables occurs

By default the **/spdata/sys1/spmon/hwevents** file is set up to do error logging and state change logging for all frames. If you are installing **splogd** on a workstation besides the control workstation in order to call your own script, you should edit the **/spdata/sys1/spmon/hwevents** file, removing the entries for SP_STATE_LOG and SP_ERROR_LOG and add a call for your own script. Refer to the **splogd** command for instructions.

The **setup_logd** command performs the following steps:

1. Creates directories in **/var/adm** that the logging daemon uses, if they do not already exist.
2. Adds an entry to **syslog.conf** for **daemon.notice** and sends a HUP signal to **syslogd** to reread its configuration file.
3. Adds errlog templates for SP messages.
4. Adds the **splogd** daemon to SRC as the **splogd** subsystem.
5. Adds an entry for **splogd** to **/etc/inittab**.

If you do not want to perform any of the preceding steps on your workstation, do not run **setup_logd**. If you are only using **splogd** to call your own script, you might only want to do step 4 and step 5 (add **splogd** to SRC and **/etc/inittab**).

To run the logging daemon on a separate workstation, you must add the following to the **/etc/environment** file:

SP_NAME={*control_workstation*}

To move a subset of error logging off of the control workstation, edit **/spdata/sys1/spmon/hwevents** on the control workstation to define the subset that you want to monitor. Then **stopsrc** and **startsrc** the logging daemon on the control workstation to reread the **hwevents** file.

Starting and Stopping the splogd Daemon

The **splogd** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **splogd** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **splogd**. In order to start the **splogd** daemon, use the **startsrc** −**s splogd** command. This starts the daemon with the default arguments and SRC options. The **splogd** daemon is setup to be respawnable and be the only instance of the **splogd** daemon running on a particular node or control workstation. Do **not** start the **splogd** daemon from the command line without using the **startsrc** command to start it.

To stop the **splogd** daemon, use the **stopsrc** −**s splogd** command. This stops the daemon and does not allow it to respawn.

To display the status of the **splogd** daemon, use the **lssrc** −**s splogd** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4 Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget** −**q "subsysname=splogd" SRCsubsys** command.

Files

/etc/inittab AIX file that contains a list of parameters to be brought up during initialization.

/spdata/sys1/spmon/hwevents
File that describes what logging is performed and what user exits are called.

/etc/syslog.conf
Describes where syslog messages are logged.

Related Information

Daemon: **splogd**

Refer to *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* for more information on setting up Hardware Monitor clients on separate workstations and the System Resource Controller.

Examples

1. To start the **splogd** daemon, enter:

   ```
   startsrc -s splogd
   ```

2. To stop the **splogd** daemon, enter:

   ```
   stopsrc -s splogd
   ```

3. To display the status of the **splogd** daemon, enter:

   ```
   lssrc -s splogd
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **splogd** daemon, enter:

   ```
   odmget -q "subsysname=splogd" SRCsubsys
   ```

setup_server

Purpose

setup_server – Configures a node or control workstation as a boot/install server.

Syntax

setup_server [–**h**]

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken.

Operands

None.

Description

Use this command to set up the node on which it is run as a boot/install server for client nodes as defined in the System Data Repository (SDR).

On a boot/install server, this command:

- Defines the boot/install server as a Network Installation Management (NIM) master
- Defines the resources needed for the NIM clients
- Defines each node that this server installs as a NIM client
- Allocates the NIM resources necessary for each NIM client
- Creates the **node.install_info** file containing netinstall information
- Creates the **node.config_info** file containing node-specific configuration information be used during network boot
- Creates server key files containing the service keys for the nodes
- Copies the install images from the control workstation for nodes which are boot/install servers

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

Creation of the Network Installation Management (NIM) lppsource resource on a boot/install server will result in **setup_server** creating a lock in the lppsource directory on the control workstation. The **setup_server** command calls **mknimres** which creates the lock.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–1 Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/setup_server

Related Information

Commands: **allnimres**, **create_krb_files**, **delnimclient**, **delnimmast**, **export_clients**, **mkconfig**, **mkinstall**, **mknimclient**, **mknimint**, **mknimmast**, **mknimres**, **setup_CWS**, **unallnimres**

Examples

To prepare a boot/install server node, enter the following on that node:

```
setup_server
```

sp_configd

Purpose

sp_configd – Starts the Simple Network Management Protocol (SNMP) SP Proxy daemon as a background process.

Syntax

sp_configd [–**T**] [–**t** *secs*] [–**s** *secs*] [–**e** *secs*]

Flags

–**T** Specifies whether to perform internal tracing. Trace entries are placed in the **/var/tmp/sp_config.log** file. The default is off.

–**t** *secs* Specifies the amount of time, data instance values that are nonconstant should be kept in cache before being considered stale. This is used to improve the performance associated with a dump of the ibmSPEMVarValuesTable (which is a series of SNMP getnext-requests). When a specific instance value from this table is requested by an SNMP get-request, the latest value is obtained from the SP Event Manager (EM) regardless of the amount of elapsed time since the last request for this data. If the –**t** flag is not specified, a default value of 720 seconds is used.

–**s** *secs* Specifies the amount of elapsed time between sending requests for the SP EM to determine the set of EM variables for which a resource monitor is currently active. Current EM resource instance values can only be obtained for those EM resource which are currently being monitored. If the –**s** flag is not specified, a default value of 1200 seconds is used.

–**e** *secs* Specifies the amount of elapsed time between retrying failed EM connection attempts. EM connection initialization causes requests to be sent to the System Data Repository (SDR) which may hinder performance if attempted too frequently. If the –**e** flag is not specified, a default value of 60 seconds is used.

Operands

None.

Description

The **sp_configd** daemon is internally configured as an SNMP Multiplexing Protocol (SMUX) peer, or proxy agent, of the **snmpd** daemon on the control workstation and on each node of the SP. For more information, refer to the “Managing SP System Events in a Network Environment” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*.

The **sp_configd** daemon provides the following functions:

- It receives requests from network monitors for data from the ibmSP MIB (these requests are routed from the **snmpd** daemon to the **sp_configd** daemon over the SMUX interface). The results are returned by the **sp_configd** daemon to

the **snmpd** daemon by the SMUX interface and are then sent to the originating monitor by the **snmpd** agent.

- It sends trap notifications about events occurring on the SP to all hosts listed in the **snmpd** daemon configuration file.

The **snmpd** daemon should be active before the **sp_configd** daemon is started. The following command activates the **snmpd** daemon:

```
startsrc -s snmpd
```

The **snmpd** daemon is controlled by the System Resource Controller (SRC) and activated whenever the system is initialized.

The **sp_configd** daemon has several sessions with the EM. These sessions are used to maintain SP EM variable instance data and information from the last trap issued associated with an SP EM event. See the **haem** command for information on starting the SP Event Manager.

The **sp_configd** daemon should be controlled using the SRC. Entering **sp_configd** at the command line is not recommended.

Manipulating the sp_configd Daemon with the System Resource Controller

The **sp_configd** daemon is a subsystem controlled by the SRC. Use the following SRC commands to manipulate the **sp_configd** daemon:

lssrc Gets the status of a subsystem, group of subsystems, or a subserver. The long status form of the **lssrc** command is not supported.

startsrc Starts a subsystem, group of subsystems, or a subserver. Issuing the **startsrc** command causes the **sp_configd** daemon to generate a coldStart trap. Use the –**a** switch to override default switch values.

stopsrc Stops a subsystem, group of subsystems, or a subserver.

Files

/etc/services Contains port assignments for required services. The following entry must be present in the **/etc/services** file if the entries are not already present:

```
smux 199/tcp
```

Notes:

1. The SMUX port must be 199.
2. The **/etc/services** file is shipped with this entry already in place.
3. If the **/etc/services** file is being served from a server, this entry must be present in the server's **/etc/services** file.

/etc/snmpd.conf

Specifies the SMUX association configuration for the **sp_configd** Proxy Agent. The following entry must be present:

```
smux    1.3.6.1.4.1.2.6.117    sp_configd_pw    # sp_configd
```

These entries are created when the SP is installed.

/etc/snmpd.peers

Specifies the configuration for the **sp_configd** SMUX peer. The following entry must be present:

```
sp_configd    1.3.6.1.4.1.2.6.117    sp_configd_pw
```

These entries are created when the SP is installed.

Security

You must have root privilege to run this command or be a member of the system group.

Related Information

See the "Understanding the SNMP Daemon" and "Problem Determination for the SNMP Daemon" chapters in *AIX Version 3.2 System Management Guide: Communications and Networks*.

See the "Understanding the Simple Network Management Protocol (SNMP)", "Understanding the Management Information Base (MIB)", and "xgmon Overview for Programmers" chapters in *AIX Version 3.2 Communication Programming Concepts*.

See the "Managing SP System Events in a Network Environment" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

1. To start the **sp_configd** daemon, enter a command similar to the following:

   ```
   startsrc -s sp_configd -a "-T"
   ```

   This command starts the **sp_configd** daemon and logs information to the **/var/tmp/sp_configd.log** file.

2. To stop the **sp_configd** daemon normally, enter:

   ```
   stopsrc -s sp_configd
   ```

   This command stops the daemon. The –**s** flag specifies the subsystem that follows to be stopped.

3. To get short status from the **sp_configd daemon**, enter:

   ```
   lssrc -s sp_configd
   ```

   This command returns the daemon name, process ID, and state (active or inactive).

sp_configdctrl script

Purpose

sp_configdctrl – A control script that is used to manage the installation of the SP Simple Network Management Protocol (SNMP) Proxy Agent subsystem.

Syntax

sp_configdctrl { –**a** | –**s** | –**k** | –**d** | –**c** | –**t** | –**o** | –**r** | –**h** }

Flags

–**a**	Adds the subsystem.
–**s**	Starts the subsystem.
–**k**	Stops the subsystem.
–**d**	Deletes the subsystem.
–**c**	Cleans the subsystem, that is, deletes it.
–**t**	Turns tracing on for the subsystem.
–**o**	Turns tracing off for the subsystem.
–**r**	Refreshes the subsystem.
–**h**	Displays usage information.

Operands

None.

Description

Use this command to install or remove the SP SNMP Proxy Agent daemon. This command can be issued only by a user with root privileges or by a member of the system group.

The **sp_configdctrl** control script controls the operation of the SP SNMP Proxy Agent subsystem. The subsystem is under the control of the System Resource Controller (SRC). The subsystem is called **sp_configd**.

An instance of the SP SNMP Proxy Agent subsystem executes on the control workstation and on every node of a system partition. Because the information about SP nodes and Event Manager (EM) variables exists in system partitions, it is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the SP SNMP Proxy Agent subsystem group is organized as follows:

Subsystem	SP SNMP Proxy Agent
Subsystem Group	None

SRC Subsystem Name

sp_configd

The **sp_configd** subsystem is associated with the **sp_configd** daemon.

The subsystem name on the nodes and the control workstation is **sp_configd**. There is one daemon per node and control workstation.

Daemons **sp_configd**

The **sp_configd** daemon provides the SP SNMP Proxy Agent function.

The **sp_configdctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **sp_configdctrl** script provides a variety of controls for operating the SP SNMP Proxy Agent subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystem, that is, deleting it from all system partitions
- Turning tracing on and off

Adding the Subsystem

When the –**a** flag is specified, the control script uses the **mkssys** command to add the SP SNMP Proxy Agent subsystem to the SRC. The control script operates as follows:

1. It makes sure that the **sp_configd** daemon is stopped.
2. It removes the **sp_configd** subsystem from the SRC (just in case it is still there).
3. It adds the **sp_configd** subsystem to the SRC.
4. It adds an entry for the **sp_configd** subsystem to the /**etc**/**inittab** file. The entry ensures that the subsystem is started during boot.
5. It adds a smux entry to the /**etc**/**snmpd.conf** file and a password entry to the /**etc**/**snmpd.peers** file for the **sp_configd** Proxy Agent if they do not currently exist.
6. It appends the ibmSP MIB definitions to the /**etc**/**mib.defs** file if they do not currently exist.
7. It issues a **refresh -s snmpd** command so that **snmpd** processes the new entries placed in the /**etc**/**snmpd.conf** and /**etc**/**snmpd.peers** files.
8. It adds an errnotify stanza for the **snmp_trap_gen** function to the Object Data Manager (ODM). This function notifies the SP SNMP Proxy Agent when an entry is written to the AIX errlog which has a template specifying **Alert = true**.

Starting the Subsystem

When the –**s** flag is specified, the control script uses the **startsrc** command to start the SP SNMP Proxy Agent subsystem, **sp_configd**.

Stopping the Subsystem

When the –**k** flag is specified, the control script uses the **stopsrc** command to stop the SP SNMP Proxy Agent subsystem, **sp_configd**.

Deleting the Subsystem

When the –**d** flag is specified, the control script uses the **rmssys** command to remove the SP SNMP Proxy Agent subsystem from the SRC. The control script operates as follows:

1. It makes sure that the **sp_configd** daemon is stopped.
2. It removes the **sp_configd** subsystem from the SRC using the **rmssys** command.
3. It removes the entry for the **sp_configd** subsystem from the /**etc**/**inittab** file.
4. It removes entries from /**etcsnmpd.conf** and /**etc**/**snmpd.peers** and removes ibmSP MIB definitions from /**etc**/**mib.defs**.

Cleaning Up the Subsystem

When the –**c** flag is specified, the control script stops and removes the SP SNMP Proxy Agent subsystem from the SRC. The control script operates as follows:

1. It stops the subsystem using the **stopsrc -s sp_configd** command.
2. It removes the subsystem from the SRC using the **rmssys** command.
3. It removes entries from /**etcsnmpd.conf** and /**etc**/**snmpd.peers** and removes ibmSP MIB definitions from /**etc**/**mib.defs**.

Turning Tracing On

When the –**t** flag is specified, the control script turns tracing on for the **sp_configd** daemon, by stopping the daemon and restarting it with the –**T** option.

Turning Tracing Off

When the –**o** flag is specified, the control script turns tracing off for the **sp_configd** daemon, by stopping the daemon and restarting it without the –**T** option.

Refreshing the Subsystem

The –**r** flag has no effect for this subsystem.

Files

/etc/snmpd.peers
Contains password entries.

/etc/snmpd.conf
Contains smux entries.

/etc/mib.defs Contains the ibmSP MIB definitions.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Security

You must be running with an effective user ID of root.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

AIX Version 4 Commands Reference

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*.

Location

/usr/lpp/ssp/bin/sp_configdctrl

Related Information

Commands: **sp_configd**

Examples

1. To add the SP SNMP Proxy Agent subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -a
   ```

2. To start the SP SNMP Proxy Agent subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -s
   ```

3. To stop the SP SNMP Proxy Agent subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -k
   ```

4. To delete the SP SNMP Proxy Agent subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -d
   ```

5. To clean up the SP SNMP Proxy Agent subsystem on all system partitions, enter:

   ```
   sp_configdctrl -c
   ```

6. To turn tracing on for the **sp_configd** daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -t
   ```

7. To turn tracing off for the **sp_configd** daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -o
   ```

8. To display the status of the SP SNMP Proxy Agent subsystem on a node or the control workstation, enter:

   ```
   lssrc -s sp_configd
   ```

spacctnd

Purpose

spacctnd – Enters accounting data into the System Data Repository for a node or group of nodes.

Syntax

spacctnd {[–**c** *acct_class_id*] | [–**e** {**true** | **false** | **default**}]
[–**j** *acct_job_charge*] [–**x** {**true** | **false**}]}
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}

Flags

–**c** *acct_class_id* Indicates that the accounting class identifier attribute of each specified node should be changed to the value of *acct_class_id*. The accounting class identifier is an arbitrary string. All nodes with the same string value constitute a class for purposes of grouping and merging accounting data.

–**e** Indicates that the accounting enabled attribute of each specified node should be changed. The accounting enabled attribute is an indicator of whether accounting is enabled for the node. The possible values are:

true Accounting is enabled
false Accounting is disabled
default Accounting is enabled based on the value of the SP accounting enabled attribute

–**j** *acct_job_charge*
Indicates that the accounting job charge value of each specified node should be changed to the value of *acct_job_charge*. The job charge value is used to determine the number of *charge fee units* to charge a user for exclusive use of the node. Its value is in units of seconds per charge fee unit. This value must be expressed as a float value with one or more digits followed by a decimal point which is followed by one or more digits.

–**x** Indicates whether accounting start and end job records and thus chargefee records are generated for jobs having exclusive use of the node. A value of **true** specifies that exclusive use accounting is enabled and start and end job records are generated. A value of **false** specifies that exclusive use accounting is not enabled and start and end job records are not generated.

–**N** *node_group* Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use

the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Indicates which frame is the starting frame for the range of nodes in this operation. If you use the *start_frame*, *start_slot*, and *node_count* fields, do not use the *node_list* field. Select a value from 1 through 64.

start_slot Indicates which slot is the starting slot for the range of nodes in this operation. The slot is the position in the rack that a node occupies. For example, for a thin node which is the second node in a rack that has a wide node in the first slot, the slot number is 3. If you use *start_frame*, *start_slot*, and *node_count*, do not use the *node_list* field. Specify the start slot as a number from 1 through 16.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Indicates which nodes are to be used for the range of nodes in this operation. If the combination of *start_slot* and *node_count* goes past the nodes in a frame, the next sequential frame is used for the operation. If you use *start_frame*, *start_slot*, and *node_count*, do not use the *node_list* field. Specify a value from 1 through 1024.

Note: The *node_count* is considered to be within the current system partition.

Description

Run this command during installation of the SP or later to set the accounting class identifier, the accounting enabled attribute, job charge value or the exclusive use accounting enabled attribute of a node or set of nodes.

You can use the System Management Interface Tool (SMIT) to run the **spacctnd** command. To use SMIT, enter:

```
smit node_data
```

and select the Accounting Information option.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

The following example adds accounting SDR information for a system with 2 frames and 32 nodes. Accounting and exclusive use accounting is to be enabled for each node and 60 seconds of exclusive use by a user is to constitute one *charge fee unit*.

```
spacctnd -e true -j 60.0 -x true 1 1 32
```

spacs_cntrl

Purpose

spacs_cntrl – Controls interactive access to SP nodes.

Syntax

spacs_cntrl [–**v** –**s**] [–**d**] [–**h**] [–**l**] [–**f** *file_name*] [–**n** *netgroup_name*]
{**allow** | **block** | **deny** | **unblock**} *user_name* ...

Flags

–**d** Specifies debugging mode. Displays additional messages, when available, to trace program flow and system errors.

–**f** *file_name* Specifies the name of a file in which the user names to be allowed or denied access are listed in a column.

–**h** Displays the usage message when present on the command line.

–**l** Specifies log messages. This flag logs messages to the **/var/adm/SPlogs/spacs/spacs.log** file. Included are all messages regarding user states that are be displayed if the –**v** flag is specified, as well as any debug messages if –**d** is specified. (This is lowercase **l**, as in **l**ist.)

–**s** Specifies suppress mode. No error messages displayed. If –**l** is specified, error messages are sent to log file.

–**v** Specifies verbose mode. A message is displayed for each user, containing the date, time and state of the user. User states resulting from this command include:

Access was removed
Access was allowed
Access removed, user allowed
Access removed, user denied
Access allowed, user allowed
Access allowed, user denied
User name is not valid or is root.

–**n** *netgroup_name*
Accepts one NIS netgroup name of a netgroup that contains user names in the user field. Netgroups embedded in a given netgroup name is resolved.

allow Used by job submission systems such as the Resource Manager. Requests that interactive access be granted to run a parallel job. Result depends on user state.

block Used by root user to set user state to a known denied state and remove user state information used by job submission systems.

deny Used by job submission systems such as the Resource Manager. Requests that interactive access be denied after running parallel job. Result depends on user state.

unblock Used by root user to set user state to a known allowed state and remove user state information used by job submission systems.

Operands

user_name Specifies the user name for which access is to be allowed or denied. Delineate with a blank space if specifying more than one user name.

Description

Use caution when issuing this command while the Resource Manager is running. If the Resource Manager is configured to use Login Control, you may cause loss of user state information. For more Login Control information, refer to *IBM Parallel System Support Programs for AIX: Administration Guide*.

The following types of access can be disallowed when **spacs_cntrl block** or **deny** is used:

login
rlogin
AIX **rsh**
AIX **rcp**
AIX **rexec**
SP **rsh**
SP **rcp**

The **spacs_cntrl** command does not allow individual types of access to be disallowed.

Duplicate user names are removed from the user list whether entered from the command line or in a file.

If you add a new user to a node for which all users are denied, you must reissue **spacs_cntrl** to deny the new user as well.

Flags and Logging

Flags specified in combination have the following results:

None	Error messages go to standard output.
–l	Error messages got to standard output and are logged.
–l –s	Error messages go to log only.
–s	Error messages suppressed.
–l –d –v –s	All messages go to log only.
–d –v –s	Messages suppressed.
–d –l	Debug and error messages go to standard output and log.

Use of the verbose flag (**–v**) causes the command to run longer due to extra processing required to find state information.

Examples

1. To block a single user (Betty) on a single parallel node, on that node enter:

   ```
   spacs_cntrl block betty
   ```

2. To block users on multiple nodes, enter:

   a. Create the **block_usr_sample** file after adjusting threshold uid.

   b. Send file to all nodes in the current system partition. Note this example would require **rsh** privileges on the nodes.

      ```
      dsh -a rcp  root@mynode:/tmp/usr.input /tmp/usr.input
      ```

   c. Issue the **spacs_cntrl** command to block users to all the nodes in the current system partition.

      ```
      dsh -a spacs_cntrl -f /tmp/usr.input block
      ```

spadaptrs

Purpose

spadaptrs – Enters configuration data for an additional adapter for a node or series of nodes in the System Data Repository (SDR).

Syntax

spadaptrs [–**s** {**yes** | **no**}] [–**r** {**4** | **16** | **autosense**}]
[–**t** {**bnc** | **dix** | **NA**}] [–**a** {**yes** | **no**}] [–**n** {**yes** | **no**}]
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}
adapter_name starting_IP_address netmask

Flags

–**s yes** | **no** Indicates whether IP addresses should be skipped when the process to assign IP addresses to a range of nodes encounters an unused slot. For example, when a node follows a wide node, if you want that node to have the next sequential IP address after the wide node, specify –**s no**. However, if you want that node to have the second sequential IP address after the wide node, specify –**s yes**. If you are specifying your nodes with a *node_list*, you must not specify –**s yes**. If you are specifying IP addresses for the High Performance Switch and you want to use switch node numbers, you must not specify –**s yes**.

If –**s** is not specified, the default is **no**.

–**r 4** | **16** | **autosense**
Specifies the token-ring network speed. This is required for **tr0** and **tr1** token-ring adapters. Specify **4** for 4MB per second, **16** for 16MB per second, and **autosense** for adapters that automatically choose the network speed.

–**t bnc** | **dix** | **NA**
Designates the Ethernet type. Use **dix** to designate a thick Ethernet (also called a twisted pair). Use **bnc** to designate a thin Ethernet. Use **NA** to designate a non or auto selectable Ethernet card. The default is **bnc**.

–**a no** | **yes** Indicates whether you want Address Resolution Protocol (ARP) to be used for the High Performance Switch. If you want to assign IP addresses freely, as for other adapters, you must specify **yes**. If you specify –**a no**, you must not specify –**n no**. Do not use this flag unless you are specifying IP addresses for the **css0** adapter. If you do not specify –**a**, the default is **no**.

–**n yes** | **no** Indicates whether you want to use switch node numbers for assigning IP addresses for the High Performance Switch. If you want to assign IP addresses freely, as for other adapters, you must specify –**n no** and –**a yes**. If you specify –**n yes**, you must not specify –**s yes**. If you specify –**n yes**, you must not specify your nodes with a node list. Do not use this flag unless you are specifying IP addresses for the **css0** adapter. If you do not use –**n**, the default is **yes**.

–**N** *node_group*
Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. This must be the lowest numbered frame which contains a node in the current system partition when you are entering data for the High Performance Switch (adapter name **css0**) and you are specifying –**n yes**.

start_slot Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive. This must be the lowest numbered slot in the lowest numbered frame in the current system partition when you are entering data for the High Performance Switch (adapter name **css0**) and you are specifying –**n yes**.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Specifies the number of nodes to be used for this operation. The information is added sequentially to nodes in slots within a frame and, if the slots in a frame are exhausted, to slots in the next sequential frame. Specify a value between 1 and 1024 inclusive. This must be the total number of nodes in your current system partition when you are entering data for the High Performance Switch (adapter name **css0**) and you are specifying –**n yes**.

Note: The *node_count* is considered to be within the current system partition.

adapter_name Specifies the name of the adapter. Specify a valid name from among: **en1**, **fi0**, **fi1**, **tr0**, **tr1**, or **css0**.

starting_IP_address
Specifies the IP address of the first node on the network. IP addresses of subsequent nodes are created via incrementing the IP address for each node.

Each IP address used in the operation must be resolved by the **host** command on the control workstation.

netmask Specifies the netmask for the network on which the adapter resides. Specify a valid IP address.

Description

Execute this command during installation of the SP to identify the IP addresses, netmask, and default route associated with node adapters other than en0. If all your IP addresses are in the same block, run this command once. If you have "holes" in your IP addressing scheme, run this command once for each block of addresses you wish to assign. Special considerations apply to how IP addressing is done for the High Performance Switch. Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for more High Performance Switch information.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spadaptrs** command. To use SMIT, enter:

```
smit node_data
```

and select the Additional Adapter Information option.

Notes:

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl** **–r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.

Related Information

Commands: **syspar_ctrl**

Examples

If you specify the **–s** flag to skip IP addresses when setting the **css0** switch addresses, you must also specify **–n no** to not use switch numbers for an IP address assignment. You must specify **–a yes** to use ARP.

```
spadaptrs -s yes -n no -a yes 1 1 30 css0 129.33.34.1 255.255.255.0
```

spapply_config

Purpose

spapply_config – Applies a system partition configuration to the SP system.

Syntax

spapply_config [–**h**] [–**v**] [–**A**] [–**F**] [–**q**] *config_dir*/*layout_dir*

Flags

–**h** Displays usage information. If this command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid options or operands are entered with the –**h** flag).

–**v** Verifies, but does not apply, the specified configuration layout. With this option, the command:

- Checks the contents of each **custom** file in the specified layout to ensure that the required stanza entries exist
- Describes which system partitions would be changed and which would be unchanged by applying the specified configuration layout
- Lists all nodes in the changed system partitions which are not shutdown

–**A** Archives the System Data Repository (SDR). With this flag, a copy of the SDR prior to applying the configuration is saved using the **SDRArchive** command.

–**F** Corrects nonfatal errors encountered in the IBM Virtual Shared Disk subsystem in the application of the specified configuration layout. Fatal errors encountered cause the command to terminate prior to applying the specified layout.

–**q** Specifies quiet mode. This option suppresses all status messages as well as the output from most internally called commands. The list of changed and unchanged system partitions, the list of nodes in changed system partitions which are not shutdown, and any warning or error messages are still displayed with this option.

Operands

config_dir Specifies the directory name for a configuration directory.

layout_dir Specifies the directory name for a layout directory within the configuration directory.

Description

This command functions in two phases: verification and application. Before applying a new system partition configuration, the administrator should back up the SP system SDR. This can be accomplished by using either the **SDRArchive** command or by using the –**A** flag on **spapply_config**. If your system has an SP switch, the **Eunpartition** command must be run before applying a new system partition configuration. Failure to do this will produce unpredictable results in the

new system partitions. Refer to the "Managing System Partitions" chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information.

The layout directory contains one system partition directory for each system partition in the configuration. Each partition directory contains the switch **topology** file and **nodelist** file. It also contains the **custom** file (created and updated by the **spcustomize_syspar** command). The **spapply_config** command verifies that these files exist. It also verifies the contents of the **custom** file. If an error is encountered in this verification phase, the command issues an appropriate message and terminates without attempting to apply a configuration layout that is not valid. As part of its verification phase, this command also calls the **verparvsd** command to determine the impact on the IBM Virtual Shared Disk subsystem of applying the specified configuration layout. If any errors or warnings are returned from **verparvsd**, the **spapply_config** command reports those messages and stops. The **–F** flag can be used to alter this behavior by correcting nonfatal IBM Virtual Shared Disk errors encountered in the analysis of the IBM Virtual Shared Disk subsystem.

As part of its processing, **spapply_config** displays to standard output the list of changed system partitions and the list of unchanged system partitions. A changed system partition is a currently-defined partition which will be changed in some way by the application of the specified configuration layout. Nodes in changed system partitions should be shutdown prior to applying that configuration. Conversely, an unchanged system partition is a currently-defined partition which will be unchanged by the application of the specified configuration layout. Nodes in unchanged system partitions can remain in operation during the application of this configuration layout. The **spapply_config** command issues the **Eannotator**, **Eprimary**, and **Etopology** commands as necessary.

The **spapply_config** command issues status messages which track the progress of operation to standard output. These messages along with the lists of changed and unchanged system partitions can be suppressed by the using the **–q** flag.

In the event that **spapply_config** encounters an error during the application phase, a descriptive error message is displayed and the command stops. In this case, it will be necessary to restore the SP SDR and the system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) to their previous state by using the **sprestore_config** command.

Note: Due to system partitioning changes, your SP_NAME environment variable may no longer be set to a valid system partition name. To get a list of valid system partition names, enter the **splst_syspars -n** command. Then verify that your SP_NAME environment variable is either unset or set to one of the system partition names in the list.

Files

nodelist Contains a list of switch node numbers contained in a system partition (used internally, not by end users).

topology Contains the wiring configuration information for switch-to-switch and node-to-switch cabling in a switch network. This information is used during switch initialization.

Related Information

Commands: **Eunpartition**, **SDRArchive**, **spcustomize_syspar**, **spdisplay_config**, **sprestore_config**, **spverify_config**, **syspar_ctrl**, **verparvsd**

Files: **nodelist**, **topology**

Examples

1. To apply the system partition configuration represented by the **config.4_12/layout.2** layout directory, enter:

   ```
   spapply_config config.4_12/layout.2
   ```

2. To check (but not apply) the system partition configuration represented by the **config.8_8/layout.1** layout directory, enter:

   ```
   spapply_config -v config.8_8/layout.1
   ```

spbootins

Purpose

spbootins – Enters boot/install configuration data for a node or series of nodes in the System Data Repository (SDR).

Syntax

spbootins {[–**n** *boot_server*] [–**i** *install_image_name*]
[–**h** {**hdisk***nn* [,**hdisk***nn*] | *location*[,*location*] |
location[:*location*]}
[–**r** {**install** | **customize** | **disk** | **maintenance** | **diag** | **migrate**}]
[–**v** *lppsource_name*]
[–**u** {*usr_server_id* | **local**}] [–**g** {*usr_gateway_id* | **0**}]
[–**a** {**en0** | **en1** | **tr0** | **tr1**}] [–**p** *PSSP_level*]} [–**s** {**no** | **yes**}]
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}

Flags

–**n** *boot_server* Identifies the boot/install server for the nodes you have specified. You can identify the boot/install server by node number, a frame, slot identifier, the host name for the node, or the IP address for an adapter on the node. The value of the boot/install server at installation depends on how many frames are in your system. In a one-frame system, the control workstation (node 0) is the default server for each node. In a multiframe system, the default server for the first node in each frame is the control workstation, and the default server for the rest of the nodes in a frame is the first node in that frame.

–**i** *install_image_name*
Specifies the name of the install image to be used for the nodes when they are next network-installed. Specify a file in the /**spdata**/**sys1**/**install.images** directory on the control workstation. At installation, the value for each node's install image name is **default**, which means that the default install image name for the system is used for each node. The default install image name for the system is found in the SP object, and its initial value is **bos.obj.ssp.41**.

–**h** Indicates the hard disks to be used for installation for the nodes specified. The **rootvg** is defined on the disks indicated, and all data on the disks is destroyed. Valid hard disk names are the disk device or the disk location. The disk device is specified by **hdisk***nn*, where *nn* is a one- or two-digit number indicating which hard disk is to be used. The disk location is specified in this format for AIX 4.1 systems 00-00-00-0,0 or in this format for an AIX 3.2.5 system 00-00-00-00. Specify one or more disks. If more than one disk is to be used for installation, separate the disk device names with a comma (for example, **hdisk0**,**hdisk1**), separate AIX 4.1 disk locations with a colon (for example, 00-00-00-0,0:00-00-00-1,0), and separate AIX 3.2.5 disk locations with a comma (for example, 00-00-00-00,00-00-00-01). The first of

the two hard disk specified will be used for the boot disk. At installation, the value for each node's **install_disk** is **hdisk0**.

–r Specifies the boot/install server's response to the bootp request from the nodes.

install Indicates that you should specify **install** if you want the server to perform a network install (overwrite install) and customize each node.

customize Indicates that you should specify **customize** if you want the server to place node-specific configuration information from the SDR into each node's local Object Data Management (ODM).

disk Indicates that you should specify **disk** if you want the server to ignore the bootp request and have each node boot from its local disk.

maintenance Indicates that you should specify **maintenance** to have each node boot in a prompted mode.

A node that boots in a prompted mode, comes up with the "Install/Maintenance" panel. From this panel, you can choose option 3 to start a limited function maintenance shell. You may access files in the root volume group (**rootvg**) by choosing the panels to mount the root volume group and enter a shell.

diag Sets the bootp_response to **diag**. The next time the node is network booted, a diagnostic menu will be displayed on the tty. From the diagnostic menu, you can execute simple or advanced diagnostics on the node or execute service aids. Service aids allow you to perform such tasks as formatting and certifying the hard drive on the node, or downloading microcode to a device attached to the node. When diagnostics are complete, set the bootp_response back to disk and reboot the node.

The **diag** parameter can be used only on the IBM Parallel System Support Programs for AIX (PSSP) Version 2 Release 1 and later nodes. PSSP Version 1 Release 2 nodes cannot run remote diagnostics.

migrate Indicates that you want the server to perform a migration installation on the specified nodes. See the *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* for more details on the migration installation method.

–v *lppsource_name*

Sets the node's lppsource name. Use this to indicate the AIX level to install on the node. The *lppsource_name* value you choose must match the directory name you choose to place the lppsource files under in the /**spdata**/**sys1**/**install** directory during installation.

See the *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* for more details.

–u Indicates the identifier for the /**usr** server for the nodes specified. The server can be a node in the SP system, the control workstation, or a workstation outside the SP system. Specify the *usr_server_id* as either a host name or an IP address. The host name or IP address must be resolved by the **host** command on the control workstation. If a node is not to be a /**usr** client, but is to have its /**usr** be local on a local disk, specify **local** for the *usr_server_id*. At installation, the value for each node's *usr_server_id* is **local**.

You can only specify this flag for a node in an IBM Parallel System Support Programs for AIX (PSSP) Version 1 Release 2 system partition.

–g *usr_gateway_id*
Indicates the gateway to be used for the /**usr** server for the nodes specified. Specify the *usr_gateway_id* as either a host name or an IP address. The host name or IP address must be resolved by the **host** command on the control workstation. If a gateway is not to be used to reach the /**usr** server, specify **0** for the *usr_gateway_id*. At installation, the value for each node's *usr_gateway_id* is **0**.

You can only specify this flag for a node in an IBM Parallel System Support Programs for AIX (PSSP) Version 1 Release 2 system partition.

–a Indicates the name of the network interface to the /**usr** server for the nodes specified. Valid names are **en0**, **en1**, **tr0**, and **tr1**. Each node in the range of nodes given must have an adapter defined of the name specified. At installation, the value for each node's *usr_client_adapter* is **en0**.

You can only specify this flag for a node in an IBM Parallel System Support Programs for AIX (PSSP) Version 1 Release 2 system partition.

–p *PSSP_level* Sets the node's PSSP level. Use this to indicate the PSSP level to install on the node. The *PSSP_level* value you choose must match the directory name you choose to place the PSSP installation files under in the /**spdata**/**sys1**/**install**/**pssplpp** directory during installation. See *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* for more details.

–s no | yes Indicates whether **setup_server** should be run on the boot servers (including the control workstation) of the indicated nodes. If you specify **–s no**, **setup_server** is not run on the node's boot server, and it must be run later to make any necessary changes to installation-related files. Specify **–s yes** if you have finished entering boot/install/usr server data during your initial installation or if you are changing data after the initial installation. Otherwise, specify **–s no**. If **–s** is not specified, the default is **–s yes**.

–N *node_group* Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

Description

Execute this command during installation of the SP, or later, to identify which node is to be the boot/install server for a node or group of nodes. You can also use this command to indicate the name of the install image to be used to install a node or group of nodes the next time they are network-installed. Each time this command is run, the **setup_server** command is run on each of the affected boot/install servers.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spbootins** command. To use SMIT, enter:

```
smit node_data
```

and select the Boot/Install Information option.

You cannot use SMIT if you are using AFS authentication services.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

1. To specify node 9 as boot/install server for nodes 10–16, enter:

   ```
   spbootins -n 9 1 10 7
   ```

2. To specify that nodes 17–32 are to have a network-install performed when they are reset, enter:

   ```
   spbootins -r install 2 1 16
   ```

spchuser

Purpose

spchuser – Changes the attributes of an SP user account.

Syntax

spchuser *attribute*=*value* ... *name*

Flags

None.

Operands

attribute=*value*	Pairs of the supported attributes and values as follows.
name	Name of the user account whose information you want to change.

Supported Attributes and Values

id	ID of the user specified by the *name* parameter.
pgrp	Principle group of the user specified by the *name* parameter.
gecos	General information about the user.
groups	The secondary groups to which the user specified by the *name* parameter belongs.
home	Host name of the file server where the home directory resides and the full path name of the directory. You can specify a host and directory in the format *host:path*, just specify the directory and have the host default to a value set in SMIT site environment panel or the **spsitenv** command, or just specify a directory and have the host default to the local machine.
login	Indicates whether the user specified by the *name* parameter can log in to the system with the **login** command. This option does not change the /**etc**/**security**/**user** file. Instead, it alters the user password field in /**etc**/**security**/**passwd**.
shell	Program run for the user specified by the *name* parameter at the session initiation.

Description

No flags are supported. Except for **home**, the rules for the supported attributes and values correspond to those enforced by the AIX **chuser** command.

You can only change the values of the supported attributes.

You can use the System Management Interface Tool (SMIT) to run the **spchuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Change/Show Characteristics of a User option.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

To change the default shell to /**bin**/**csh**, and change the secondary group membership to **dev** and **dev2** for the user account **charlie**:

```
spchuser groups=dev,dev2 shell=/bin/csh charlie
```

spcustomize_syspar

Purpose

spcustomize_syspar – Enters or verifies customization information to be used in creating a system partition.

Syntax

spcustomize_syspar [–**h**]
[–**n** *syspar_name* | *IP_address*]
[–**l** *PSSP_code_level*]
[–**d** *default_install_image* | **default**]
[–**e** *primary_node* | **default**]
[–**b** *backup_primary_node* | **default**]
config_dir|*layout_dir*|*syspar_dir* |
fully_qualified_path_name

Flags

–**h** Displays usage information.

–**n** *syspar_name* | *IP_address*
Specifies the system partition name (the control workstation host name or host name alias) or IP address (which corresponds to the system partition name) associated with this system partition.

–**l** *PSSP_code_level*
Specifies the IBM Parallel System Support Programs for AIX (PSSP) code level for the system partition.

–**d** *default_install_image* | **default**
Specifies the default install image for the system partition or **default** to direct the system to use the system-wide default install image. Refer to *IBM Parallel System Support Programs for AIX: Installation and Migration Guide* for additional information on the default install image.

–**e** *primary_node* | **default**
Specifies the primary node number for switch operations or **default** to direct the system to automatically set the default which is the first node in the node list.

–**b** *backup_primary_node* | **default**
Specifies the primary backup node number for switch operations or **default** to direct the system to automatically set the default which is the last node in the node list. This flag is valid only on SP Switch systems.

Operands

config_dir Specifies the directory name for a configuration directory.

layout_dir Specifies the directory name for a layout directory within the configuration directory.

syspar_dir Specifies the directory name for a system partition directory within the layout directory.

fully_qualified_path_name
Specifies the fully qualified path name to a system partition directory.

Description

Use this command to customize a system partition customization file (**custom**) or to display the previously-entered customization information.

For a specified system partition, the customization data can be entered with the optional parameters. If the **custom** file does not exist, you can create one by specifying the –**n** and –**l** flags. The –**d** and –**e** flags are optional when creating a **custom** file. If –**d** and –**e** are not specified, the system automatically specifies **default** to set the default install image and primary node in the newly-created **custom** file. Once the **custom** file is created, any combination of the optional parameters can be used to update the contents of the file.

If none of the optional parameters are specified with the **spcustomize_syspar** command, the contents of the customization file for the specified system partition are displayed to standard output as in the **spdisplay_config** command with the –**c** flag.

Related Information

Commands: **spapply_config**, **spdisplay_config**, **spverify_config**

Files: **nodelist**, **topology**

Examples

1. To display the customization information for the specified system partition, enter:

```
spcustomize_syspar config.4_12/layout.1/syspar.1

syspar-name:              my-partition-1
IP-address:               9.102.55.301
PSSP-code-level:          PSSP-2.1
default-install-image:    bos.4.1.2
primary-node:             9
backup-primary-node:      16
```

2. To modify the system partition name, PSSP code level, and primary node information for the specified system partition, enter:

```
spcustomize_syspar -n my-new-partition-name -l PSSP-2.1 \
-e 7 config.4_12/layout.1/syspar.1
```

3. To use the default primary node information for the specified system partition, enter:

```
spcustomize_syspar -e default config.4_12/layout.1/syspar.1
```

spcw_addevents

Purpose

spcw_addevents – Identifies the High Availability Cluster Multiprocessing (HACMP) event scripts supplied by the High Availability Control Workstation (HACWS) to the AIX High Availability Cluster Multi-Processing/6000 (HACMP/6000) software.

Syntax

spcw_addevents

Flags

None.

Operands

None.

Description

HACWS customizes the recovery of control workstation services by providing HACMP event scripts, which get executed by the HACMP/6000 software. The **spcw_addevents** command is a shell script which identifies the HACMP event scripts to HACMP/6000, without requiring the system administrator to go through all the equivalent HACMP SMIT panels.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Security

You must have root privilege to run this command.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the HACWS option.

Location

/usr/sbin/hacws/spcw_addevents

spcw_apps

Purpose

spcw_apps – Starts or stops control workstation applications in a High Availability Control Workstation (HACWS) configuration.

Syntax

spcw_apps {–**u** | –**d**} [–**i** | –**a**]

Flags

–**u** Starts control workstation applications on the local host.

–**d** Stops control workstation applications on the local host.

–**i** Sets the local host to be the inactive control workstation before starting or after stopping control workstation applications.

–**a** Sets the local host to be the active control workstation before starting or after stopping control workstation applications.

Operands

None.

Description

The control workstation services are started at boot time on a regular control workstation via entries in /**etc**/**inittab**. An HACWS configuration requires the capability to stop control workstation services on one control workstation and restart them on the other. The **install_hacws** command removes most of the control workstation entries from /**etc**/**inittab**, and the **spcw_apps** command is provided as a means to stop and start control workstation services in the HACWS configuration. In addition, the **spcw_apps** command can be used to make the inactive control workstation act as a client of the active control workstation in order to keep the two control workstations synchronized.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP) will start or stop the control workstation applications during a fail over or reintegration. The administrator should not normally have to start or stop the control workstation applications.

Exit Values

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the HACWS option.

Location

/usr/sbin/hacws/spcw_apps

Related Information

Command: **install_hacws**

Examples

In the following example, assume that the primary control workstation is currently the active control workstation. This means that the primary control workstation is providing control workstation services to the SP system. When a control workstation failover occurs, the AIX High Availability Cluster Multi-Processing/6000 (HACMP/6000) software moves the control workstation network and file system resources from the primary to the backup control workstation. In addition, control workstation applications must be stopped on the primary and restarted on the backup. HACWS provides the **spcw_apps** command to HACMP/6000 as the method to accomplish this. The HACMP/6000 software issues the following command on the primary:

```
spcw_apps -di
```

This command stops control workstation services on the active primary and then sets the primary to be the inactive control workstation. Next, the HACMP/6000 software issues the following command on the backup:

```
spcw_apps -ua
```

This command sets the backup to be the active control workstation and then starts the control workstation services on the backup. Finally, the HACMP/6000 software issues the following command on the primary:

```
spcw_apps -u
```

This command configures the primary to be a client of the backup (which is active) control workstation.

spdeladap

Purpose

spdeladap – Removes configuration data for adapters for a node or series of nodes from the System Data Repository (SDR).

Syntax

spdeladap {*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*} *adapter_name*

Flags

–**N** *node_group* — Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* — Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame — Frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot — Slot number of first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count — Number of nodes to be used for this operation. The adapter information is deleted for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

adapter_name — Name of the adapter. Specify a valid name from among: **en1**, **fi0**, **fi1**, **tr0**, **tr1**, or **css0**.

Description

Use this command to remove configuration data for adapters for a node or series of nodes from the SDR. You cannot use this command to delete data for the **en0** adapter. If you want to remove configuration data for the **en0** adapter, you should use the **spdelnode** command.

You can use the System Management Interface Tool (SMIT) to run the **spdeladap** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Adapter Information option.

Notes:

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl** –**r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.

Related Information

Commands: **syspar_ctrl**

Examples

This example deletes **tr0** adapter information for the the first two nodes in frame 2:

```
spdeladap 2 1 2 tr0
```

spdelfram

Purpose

spdelfram – Removes configuration data for a frame or series of frames from the System Data Repository (SDR).

Syntax

spdelfram [–**G**] *start_frame frame_count*

Flags

–**G** Allows for the deletion of a frame or frames which contain nodes that are outside of the current system partition.

Operands

start_frame Specifies the frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive.

frame_count Specifies the number of frames to be used for this operation. Specify a value between 1 and 64 inclusive.

Description

Execute this command to remove configuration data for a frame or series of frames from the SDR. Any node information for nodes on the frames is also removed, as well as adapter information for any of the nodes on the frames. All the nodes on all the frames to be deleted must be shut down. A frame containing a node acting as a server for another node on a frame not being removed with this operation cannot be removed with this command. In order to remove a frame containing such a node, you must configure a different boot/install or /**usr** server for the client nodes on the other frames.

You must specify the –**G** flag if you are deleting one or more frames which contain nodes outside of the current system partition. Otherwise, the command will not delete any frame.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spdelfram** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Frame Information option.

Notes:

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. You should stop the Resource Manager before the command and start it again afterward.

Examples

This example deletes the last two frames in a four-frame system:

```
spdelfram 3 2
```

spdelnode

Purpose

spdelnode – Removes configuration data for a node or nodes from the System Data Repository (SDR).

Syntax

spdelnode {*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}

Flags

–**N** *node_group* Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

Description

Execute this command to remove configuration data for a node or series of nodes from the SDR. Any adapter information associated with the node is also removed. All the nodes to be deleted must be shut down. A node acting as a server for another node cannot be removed with this command. In order to remove a node which is a server, you must configure a different boot/install server for the client nodes.

This command can be used to delete nodes in the current system partition. To delete nodes in another system partition, you must make it your current system partition.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spdelnode** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Node Information option.

Notes:

1. This command should be run only on the control workstation.
2. After running this command, you must issue the **syspar_ctrl** **-r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.

Security

You must be logged into the control workstation as root to run this command.

Related Information

Commands: **syspar_ctrl**

Examples

1. This example deletes the first two nodes in frame 2:

   ```
   spdelnode 2 1 2
   ```

2. To delete the nodes in the node group temp_nodes, enter:

   ```
   spdelnode -N temp_nodes
   ```

spdisplay_config

Purpose

spdisplay_config – Displays system partition configuration information which can be used to partition an SP system.

Syntax

spdisplay_config [–**h**] [–**R**] [–**d**] [–**c**] [–**n**] [*config_dir* [/*layout_dir* [/*syspar_dir*]] | *fully_qualified_path_name*]

Flags

–**h** Displays usage information. If this command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid options are entered with the –**h** flag).

–**R** Recursively displays information for all levels below the specified directory level.

–**d** Displays the description file (**layout.desc**) for the specified layout. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a layout directory.

–**c** Displays the customization file (**custom**) for the specified system partition. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a system partition directory.

–**n** Displays the node list file (**nodelist**) for the specified system partition. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a system partition directory.

Operands

config_dir Specifies the directory name for a configuration directory.

layout_dir Specifies the directory name for a layout directory.

syspar_dir Specifies the directory name for a system partition directory.

fully_qualified_path_name
Specifies the fully qualified directory path for a configuration directory, layout directory, or system partition directory.

Description

This command displays system partition information stored in the system partition information directory structure. Depending on the option and operand specified, the information displayed is at the configuration, layout, or system partition level. The output of this command is normally restricted to the SP on which it is executed. To display information for configurations applicable to SP systems other than the one on which it is executed, a fully qualified path name must be provided. This command does not display current system partition information (that function is provided with the **splstdata** command). If the command is issued without specifying a directory path name, the list of all valid configuration directories for the SP on which the command is running is displayed. If none of the file option flags

(–**c**, –**d**, –**n**) are entered, the names of the files and subdirectories located on (and all levels below if –**R** is specified) the specified directory are displayed. If any of the file option flags are entered, the contents of the requested files are displayed instead.

Files

custom	Contains customization information for a partition (such as, partition name, IP address, code level, and so on).
layout.desc	Describes the node slot breakdown for a partitioning configuration (which nodes in which partition).
nodelist	Contains a list of switch node numbers contained in a system partition (used internally, not by end users).
topology	Contains the wiring configuration information for switch-to-switch and node-to-switch cabling in a switch network. This information is used during switch initialization.

Related Information

Commands: **spapply_config**, **spcustomize_syspar**, **splstdata**, **spverify_config**

Files: **nodelist**, **topology**

Examples

1. To display all valid configurations for this SP, enter:

   ```
   spdisplay_config
   config.2_14
   config.4_12
   config.8_8
   ```

2. To display the list of layout directory names for a specific configuration directory to standard output, enter:

   ```
   spdisplay_config config.4_12
   layout.1
   layout.2
   layout.3
   ```

3. To display the name of the file describing the layout and the list of system partition directory names for a specific layout directory to standard output, enter:

   ```
   spdisplay_config config.4_12/layout.2
   layout.desc
   syspar.1
   syspar.2
   ```

4. To display the list of files located in a specific system partition directory to standard output, enter:

   ```
   spdisplay_config config.4_12/layout.2/syspar.1
   custom
   nodelist
   topology
   ```

   If the –**c** flag is also supplied, only the customization information is displayed for that system partition. For example:

```
spdisplay_config -c config.4_12/layout.2/syspar.1
custom:
syspar-name:              my-partition-1
IP-address:               9.102.55.301
primary-node:             9
default-install-image:    bos.4.1.2
PSSP-code-level:          PSSP-2.1
```

If the –**n** flag is supplied, only the list of nodes is displayed for that system partition. For example:

```
spdisplay_config -n config.4_12/layout.2/syspar.1
nodelist:
switch node numbers:      4 5 6 7 8 9 10 11 12 13 14 15
node numbers:             5 6 7 8 9 10 11 12 13 14 15 16
```

All of the commands can be issued with the –**R** flag to recursively display the information on the specified directory and all the levels below it.

To display the entire system partition information directory structure for the **config.4_12** configuration on this SP, enter:

```
spdisplay_config -R config.4_12
layout.1:
layout.1/layout.desc
layout.1/syspar.1:
layout.1/syspar.1/custom
layout.1/syspar.1/nodelist
layout.1/syspar.1/topology
layout.1/syspar.2:
layout.1/syspar.2/custom
layout.1/syspar.2/nodelist
layout.1/syspar.2/topology
layout.2:
layout.2/layout.desc
layout.2/syspar.1:
layout.2/syspar.1/custom
layout.2/syspar.1/nodelist
layout.2/syspar.1/topology
layout.2/syspar.2:
layout.2/syspar.2/custom
layout.2/syspar.2/nodelist
layout.2/syspar.2/topology
```

Another example to recursively display all of the customization information for the **config.4_12** on this SP follows:

```
spdisplay_config -R -c config.4_12
layout.1/syspar.1/custom:
syspar-name:               my-partition-1
IP-address:                9.102.55.301
primary-node:              9
default-install-image:     bos.4.1.2
PSSP-code-level:           PSSP-2.1

layout.1/syspar.2/custom:
syspar-name:               my-partition-2
IP-address:                9.102.55.302
primary-node:              9
default-install-image:     bos.4.1.2
PSSP-code-level:           PSSP-2.1

layout.2/syspar.1/custom:
syspar-name:               my-partition-1
IP-address:                9.102.55.501
primary-node:              9
default-install-image:     bos.4.1.2
PSSP-code-level:           PSSP-2.1

layout.2/syspar.2/custom:
syspar-name:               my-partition-2
IP-address:                9.102.55.502
primary-node:              9
default-install-image:     bos.4.1.2
PSSP-code-level:           PSSP-2.1
```

5. To display all valid configurations for an SP, specify the fully qualified path name to its system partition information directory. For example:

```
spdisplay_config /spdata/sys1/syspar_configs/1nsb0isb
config.16
config.4_12
config.4_4_4_4
config.4_4_8
config.8_8
```

spethernt

Purpose

spethernt – Enters configuration data for a node or series of nodes in the System Data Repository (SDR).

Syntax

spethernt [–**s** {**yes** | **no**}] [–**t** {**bnc** | **dix**}]
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}
starting_IP_address netmask default_route

Flags

–**s yes** | **no** — Indicates whether IP addresses should be skipped when unused slots are encountered. For example, when a node follows a wide node, if you want that node to have the next sequential IP address after the wide node, specify –**s no**. If you want that node to have the second sequential IP address after the wide node, specify –**s yes**. If you do not specify –**s**, the default is –**s no**.

If you are specifying the nodes to be used in this operation with a *node_list* via the –**l** flag, you must not specify –**s yes**.

–**t bnc** | **dix** — Designates the Ethernet type. Use **dix** to designate a thick Ethernet (also called a twisted pair). Use **bnc** to designate a thin Ethernet. The default is **bnc**.

–**N** *node_group* — Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* — Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame — Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot — Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count — Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

starting_IP_address
Specifies the IP address or host name of the first node in this operation. IP addresses of subsequent nodes are created by incrementing the IP address for each node, depending on how the –**s** flag is set. Specify a valid IP address or host name.

Ensure that the combination of the starting IP address, the node count operand, and the –**s** flag do not result in the incrementing of the IP address to an invalid IP address.

Each IP address used in the operation must be resolved by the **host** command on the control workstation.

netmask Specifies the netmask for the en0 network. Specify a valid IP netmask.

default_route Specifies the IP address or host name of the default route for the en0 adapter network. Specify a valid IP address or host name.

Description

Execute this command during installation of the SP to identify the IP addresses, netmask, and default route associated with the en0 adapters of the nodes. If all your IP addresses are in the same block, run this command once. If you have "holes" in your IP addressing scheme, run this command once for each block of addresses you wish to assign.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spethernt** command. To use SMIT, enter:

```
smit node_data
```

and select the SP Ethernet Information option.

Notes:

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl** –**r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.

Related Information

Commands: **syspar_ctrl**

Examples

The following example adds SDR information for an en0 network of 15 nodes (frame 1 slot 1 to frame 1 slot 16 with the first node being a wide node and the rest of the nodes thin nodes all in a single system partition) with IP addresses from 129.33.32.1 to 129.33.32.15, a netmask of 255.255.255.0, and a default route of 129.33.32.200. The addresses are to be assigned to correspond with the nodes in the frame; for example, they do not increment the IP address of a wide node by 2 before assigning the IP address of the next node.

```
spethernt -s no 1 1 15 129.33.32.1 255.255.255.0 129.33.32.200
```

spevent

Purpose

spevent – Directly invokes the Event Management window of the SP Perspectives graphical user interface (GUI).

Syntax

spevent [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[{–**backgroundColor** | **bg**} *colorName*]
[{–**foregroundColor** | **fg**} *colorName*] [–**fontFamily** *name*]
[–**fontSize** *Size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to launch the Event window of the SP Perspectives GUI. The Event Perspective is a graphic vehicle for managing events and event definitions in the system.

This perspective allows the user to interact with the Event Manager and the Problem Manager. The user can create event definitions, register or unregister event definitions, and delete event definitions. A notebook is provided for viewing and editing the condition, the resource identifier, and other attributes of an event's definition. The notebook also provides users with the capability for creating new conditions. Event definitions are viewed and manipulated within system partition boundaries.

By default, the Event Definition pane displays all the event definitions with the Kerberos principal of the user within the current system partition. The user can use the filter function provided on the pane to expand or further confine the event definitions being displayed.

When the command is invoked, preferences which define the look and layout of the spevent window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

The Users Preferences are read from and saved to **$HOME/.spevent(User Profile Name)**.

The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spevent(System Profile name)**.

Restrictions

Any user may run spevent. Many actions require root privilege to run.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For information on using the Event window and the SP Perspectives GUI, please see the online help and the “Using the SP Perspectives GUI” chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*.

For information on the Event Management services, refer to “The Event Management Subsystem” and “Using the Problem Management Subsystem” chapters in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/spevent

Related Information

You can also access the Event window by using the SP Perspectives Launch Pad. The **perspectives** command invokes the launch pad. Other Perspectives windows may be launched by invoking the following commands: **sphardware**, **spsyspar**, **spvsd**, and **spperfmon**.

Examples

1. To invoke the spevent window, enter:

   ```
   spevent
   ```

2. To bring up the Event window using all the default preferences, enter:

   ```
   spevent -noProfile
   ```

spframe

Purpose

spframe – Enters configuration data for a frame or series of frames and, optionally, set up the initial System Data Repository (SDR).

Syntax

spframe [–**r yes** | **no**] *start_frame frame_count starting_tty_port*

Flags

–**r no** | **yes** — Indicates whether you want to initialize the System Data Repository. If this is the last or only time you are invoking this command during installation, specify –**r yes**. If – **yes** is specified, the /**spdata**/**sys1**/**spmon**/**hmacls.file** has the default entries created.

The default is –**r no**.

Operands

start_frame — Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

frame_count — Specifies the number of frames to be used for this operation. Specify a value between 1 and 64 inclusive.

starting_tty_port — Specifies the device special file name of the tty port to be assigned to the first frame on this operation. tty ports for subsequent frames are assigned by incrementing the tty number for each frame. Specify the full path of a valid tty device special file.

Description

Execute this command during installation of the SP to identify the tty ports to which your frames are connected. If the tty special files for your frames are consecutively numbered, run this command once during installation, specifying –**r yes**. If you have “holes” in your tty special file numbering scheme, run this command once for each block of ttys you wish to assign, specifying –**r yes** only on the last invocation of the command.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

You can use the System Management Interface Tool (SMIT) to run the **spframe** command. To use SMIT, enter:

```
smit enter_data
```

and select the Frame Information option.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

The following example enters information for four frames (frame 1 to frame 4) and indicates that frame 1 is connected to /**dev**/**tty1**, frame 2 to /**dev**/**tty2**, and so on. The System Data Repository is to be initialized with this invocation of **spframe**.

```
spframe -r yes 1 4 /dev/tty1
```

spget_syspar

Purpose

spget_syspar – Returns the IP address or name of the current system partition.

Syntax

spget_syspar [–**n**] [–**d**]

Flags

–**n** Returns a host name instead of an address.

–**d** Returns the name or IP address of the default system partition rather than the current system partition.

Operands

None.

Description

Use this command to display to standard output the IP address (in dotted decimal format) of the current or default system partition. The current system partition indicates the system partition to which System Data Repository (SDR) client requests are directed. The result is displayed in dotted decimal format unless –**n** is specified.

Restrictions

The –**d** flag will not work if the command is not issued on a control workstation or SP node.

Examples

1. To display the IP address associated with the current system partition, enter:

   ```
   spget_syspar
   ```

   You should receive output similar to the following:

   ```
   129.40.127.122
   ```

2. To display the name (host name alias of the control workstation) of the current system partition, enter:

   ```
   spget_syspar -n
   ```

   You should receive output similar to the following:

   ```
   k47sp1
   ```

sphardware

Purpose

sphardware – Directly launches the Hardware window of the SP Perspectives graphical user interface (GUI).

Syntax

sphardware [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[{–**backgroundColor** | **bg**} *colorName*]
[{–**foregroundColor** | **fg**} *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to launch the Hardware window of the SP Perspectives GUI. From the Hardware window, the user can monitor and manipulate objects within the SP system. The SP objects included in this window are the control workstation, SP system and system partitions, nodes, node groups, and frames and switch boards.

By default, the Hardware window will display the CWS/System/Syspars Work Area and the Node Work Area. The CWS/System/Syspars pane contains all system partitions and indicates the current system partition, and the Node pane contains all the nodes in the current system partition in the "frame" display control orientation.

The Node Group and Frame/Switchboard panes can be added to the Hardware window by using combo boxes at the top of the window. Likewise, any pane that is displayed can be deleted from the window by using the Delete combo box at the top of the window.

When the command is invoked, preferences which define the look and layout of the sphardware window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

The Users Preferences are read from and saved to **$HOME/.sphardware(User Profile Name)**.

The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.sphardware(System Profile name)**.

Restrictions

Any user can run the **sphardware** command. Many actions require root privilege to run.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For information on using the Hardware window and the SP Perspectives GUI, please see the online help and the "Using the SP Perspectives GUI" chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/sphardware

Related Information

The Hardware window can also be accessed by using the SP Perspectives Launch Pad. The **perspectives** command invokes the launch pad. Other Perspectives windows can be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, **spsyspar**, and **spvsd**.

Examples

1. To invoke the **sphardware** window, enter:

   ```
   sphardware
   ```

2. To force **sphardware** to display text in chartreuse, regardless of what is set in the preference files, enter:

   ```
   sphardware -foreground chartreuse
   ```

3. To start the **sphardware** window in the background, enter:

   ```
   sphardware &
   ```

sphostnam

Purpose

sphostnam – Enters host name configuration data for a node or series of nodes in the System Data Repository.

Syntax

sphostnam [–**a** *adapter_name*] [–**f** {**long** | **short**}]
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}

Flags

–**a** Indicates the adapter to be used to derive the host name for the nodes specified. If –**a** is not specified, the default is **en0**.

–**f** Specifies which form of the host name is to be used. Specify **long** if you want the host name to be the fully qualified host name and **short** if you want to use the short form of the host name. If –**f** is not specified, the default is **long**.

–**N** *node_group* Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted the operation continues in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

Description

Execute this command during installation of the SP to specify the adapter type to be used to determine the host name by which your nodes are known. You can also use this command to indicate whether you want the long (fully qualified) or short form of a host name to be used.

You can use the System Management Interface Tool (SMIT) to run the **sphostnam** command. To use SMIT, enter:

```
smit node_data
```

and select the Hostname Information option.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

The following example selects the **css0** adapter for the host name for a system with two frames and 32 nodes. The long form of the host name is to be used.

```
sphostnam -a css0 1 1 32
```

sphrdwrad

Purpose

sphrdwrad – Obtains hardware Ethernet addresses for SP nodes so they can be written to the System Data Repository.

Syntax

sphrdwrad {*start_frame start_slot* {*node_count* | **rest**} |
| –**N** *node_group* | –**l** *node_list*}

Flags

rest Indicates that, beginning with the node determined by *start_frame* and *start_slot*, all the rest of the nodes should be used for this operation.

| –**N** *node_group* Specifies a node group to be used for this operation. This node
| group must be bound to the current system partition.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

Operands

start_frame Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

start_slot Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: The *start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted the operation continues in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Note: The *node_count* is considered to be within the current system partition.

Description

Execute this command only at installation or when adding new frames or nodes. The **spframe** command must be run **before** this command so that frame information is already in the System Data Repository.

If you know your hardware Ethernet addresses, you can speed this process by putting the addresses in /**etc**/**bootptab.info**, as follows:

Create a file named /**etc/bootptab.info** (if it does not already exist), listing your SP nodes by node number (or frame, slot) followed by a blank and the hardware Ethernet address. The first token represents the node and the second token represents the hardware address. The file should look similar to this:

```
17 02608C2E48D9
19 02608C2D6712
21 02608C2E49A4
23 02608C2E48E2
```

If you do not know your hardware Ethernet addresses, use the **sphrdwrad** command to find them.

Notes:

1. The nodes should be physically powered on (but logically powered off) when you run this command.
2. The LEDs change values while this command is running.
3. You should not have a tty open to any of the nodes to be used for this command.
4. If the addresses are not found in /**etc/bootptab.info**, the **sphrdwrad** command takes a few minutes to run, and the addresses are obtained from the nodes in parallel.
5. Any nodes specified will be powered off to acquire the Ethernet addresses. The nodes remain in the powered off state, even after the addresses are received.
6. If you are adding a node, only the new node needs to be specified because any selected nodes will be powered off.
7. To avoid possible file system corruption, you should always shut down a node cleanly before powering it off. You can do this by using the **cshutdown** command or by using the SHUTDOWN/POWER OFF option in the System Monitor Graphical User Interface.

You can use the System Management Interface Tool (SMIT) to run the **sphrdwrad** command. To use SMIT, enter:

```
smit enter_data
```

and select the Get Hardware Ethernet Addresses option.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

To obtain Ethernet addresses for a new frame containing 8 nodes (4 wide nodes and 4 thin nodes), enter:

```
sphrdwrad 2 1 8
```

You should receive output similar to the following:

```
Acquiring hardware Ethernet address for node 17
Acquiring hardware Ethernet address for node 19
Acquiring hardware Ethernet address for node 21
Acquiring hardware Ethernet address for node 23
Acquiring hardware Ethernet address for node 25
Acquiring hardware Ethernet address for node 26
Acquiring hardware Ethernet address for node 27
Acquiring hardware Ethernet address for node 28
Hardware ethernet address for node 17 is 02608C2D481C
Hardware ethernet address for node 19 is 02608C2D78DF
Hardware ethernet address for node 21 is 02608C2D93B3
Hardware ethernet address for node 23 is 02608C2D8C3C
Hardware ethernet address for node 25 is 10005AFA22B9
Hardware ethernet address for node 26 is 10005AFA230A
Hardware ethernet address for node 27 is 10005AFA2229
Hardware ethernet address for node 28 is 10005AFA2210
```

splm

Purpose

splm – Views, gathers, archives, or collects log and system data.

Syntax

splm [–**a archive**] [–**cny**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**t** *table*]

splm [–**a archive**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**n**] [–**r**] [–**t** *table*] [–**y**]

splm [–**a check**] [–**h**] [–**n**] [–**t** *table*]

splm [–**a gather**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**k** *type*] [–**l cfs**] [–**o** *loc*] [–**r**] [–**s**] [–**t** *table*] [–**y**]

splm [–**a service**] [–**cny**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**p** *ipts*] [–**t** *table*]

splm [–**a service**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**n**] [–**r**] [–**t** *table*] [–**y**]

splm [–**a view**] [–**h**] [–**n**] [–**t** *table*]

Flags

The **splm** command requires the –**a** flag and an argument to select a function to execute. It also requires a log table that contains records specifying target nodes and files or commands.

–**a** *action* Specifies the function to perform: **archive**, **check**, **gather**, **service**, or **view**.

–**c** Creates a compressed tar file. For the **archive** function, this will be a tar starting the –**d** *dir* flag. For **service** collections, /**usr**/**sbin**/**snap** –**c** –**d** *dir* will be called to create the tar file. The tar file will be named **node.tar.Z**.

–**d** *dir* Specifies the path where the **archive** or **service** collection will be stored on each node. The **archive** default is /**var**/**adm**/**archives**. The **service** default is /**tmp**.

–**f** *fanout* Sets the maximum fanout value which determines how many nodes will execute in parallel. The default is 32. Any number between 1–32 can be used.

–**h** Displays usage information.

–**k** *type* For the **gather** function only, this flag indicates whether a **service** collection or **archive** is being collected.

–**l cfs** Specifies the path on the local node where the **archive** or **service** collections should be gathered.

–**n** Ignores the node designation in the input table and executes all entries on the local node only.

–**o** *loc* Specifies the device or mail location where to direct tar files.

–**p** *opts* Accepts a string of characters representing option parameters for calling snap collection. Each character relates to a category of system data to be collected. Valid characters are: a, A, D, f, g, G, k, l, n, p, s, S, t.

–**r**	Removes **archive** or **service** on each node. Exclusive with –**s** flag.
–**s**	Staggers collection to a mail location or device.
–**t**	Specifies the input table of nodes and commands.
–**y**	Appends the *yymmdd* timestamp subdirectory to the per node directory.

Operands

None.

Description

Use this command to execute a number of log management functions on a single host or a set of hosts in parallel. The **splm** functions are driven by a log table file that contains the target node designations and associated files, and the commands to execute. The table format follows:

```
# Comment line
<target nodes>:  <file or command>
```

The target node portion of a table stanza is a comma-delimited list with no blanks. The values in the list are interpreted as:

1. If the value begins with a slash (/), it is a file containing a list of node names, one per line.
2. If the value is an exclamation point (!), it refers to the local host.
3. Any string not matching 1 or 2, is interpreted as a node name.

The –**n** flag ignores the target node portion of the table and only executes on the local node. The file or command portion of the stanza specifies either a command to execute that displays information to standard output, or a file that will be archived, collected, or viewed. File specification can take advantage of Tcl file globbing (similar to csh file globbing). If the file or command portion of the stanza refers to a command to be executed, it must be followed by a redirection and a path or file name. The information generated by the command will be redirected to the path or file name under the –**d** top level directory. Use > or >> following the command to redirect the output. The view option ignores the file or command destinations and displays the file's contents or command output to the executing node.

1. To specify the local node, nodes listed in the /**tmp**/**nodelist** file and node k47n10, and archive or collect errpt output from those nodes to the **errpt.output** file under the top level directory, enter:

   ```
   !,/tmp/nodelist,k47n10:  /bin/errpt -a > errpt.output
   ```

2. To archive or collect /**etc**/**filesystems** file to a subdirectory on nodes k47n10 and k47n15, enter:

   ```
   k47n10,k47n15: /etc/filesystems  etc/filesystems
   ```

   This copies the file to the /**etc** subdirectory under the –**d** top level directory.

   Note: The –**d** top level directory is always appended with a subdirectory named **arch**_*table_name* for archives, or **srvc**_*table_name* for service collections.

splm Functions

Archive: The **archive** function copies files and redirects command output as specified in the input table to the top level directory on each node. The –**c** flag then creates a compressed tar file of the data named /**topdirectory**/*node_name***.tar.Z.** The –**r** flag removes an archive by removing all files starting from the top level directory down.

Check: The **check** function can be used to check a table for errors.

Gather: The **gather** function moves **archive** or **service** tar files to a central location on the executing node. The –**r** option removes the **archive** or **service** collection on each remote node only after the tar file was successfully copied to the central location. If the *node***.tar.Z** file is not found, the **gather** function will attempt to create one. Gathered tar files can be mailed or copied to a tape or disk device using the –**o** flag. If mailed, the files are first uuencoded. The –**l** flag specifies the file system on the local node where the tar files are to be gathered. The –**l** flag must be specified if the –**s** stagger flag is not used. The **gather** function makes two passes, if necessary. On the first pass, it allows each node to take up an equal amount of the central file system. If any nodes fail, the **gather** function retries those nodes, one at a time, until the file system is full or all the nodes are copied. If **gather** fails on any node, but a *node***.tar.Z** file exists for that node in the central location, it is moved to *node***.tar.Z.old**, and not sent to the output location. The –**s** stagger flag forces the fanout to 1, gathers the tar files one at a time, attempts to send the tar to the output location, then removes it from the local node. The –**r** flag cannot be used with –**s**. The default central location directory for stagger is /**tmp**.

Service: The **service** function first calls the AIX **snap** command to gather system data to the top level directory if the –**p** flag is used. The **snap** command creates a set of subdirectories based on the –**p** arguments. The additional data defined in the table data is then collected under the “other” subdirectory created by **snap**. If the –**p** flag is not used, the data will still be collected under the “other” subdirectory. If the –**c** flag is used, **splm** uses the **snap** –**c** command to create the **tar.Z** file. The –**r** flag can be used to remove service collections. **splm** calls **snap** –**r** which removes the tar file and all files under each **snap** subdirectory.

View: The **view** function displays the output of the command or contents of file entries in the input table to the local host.

Files

/etc/splm.allow
Restricts table commands that can be executed.

/etc/logmgt.acl
Acl file for **archive**, **gather**, and **service** functions.

/spdata/sys1/logtables/*
Contains sample tables for service collections.

Security

The **archive**, **gather**, and **service** functions of **splm** require that you have a Kerberos principal defined in the /**etc/logmgt.acl file**. The command then runs as root on all target nodes. The viewing function requires that you be Kerberos authenticated to a valid user ID on the nodes that you are executing on. The server switches IDs from root to your authenticated ID before executing.

Related Information

AIX commands: **compress snap**, **tar**, **uuencode**

The *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

1. To create an **archive** based on the entries in the /**etc/tables/logs.tab** table and to create a compressed tar file and have the **archive** under directory /**var/adm/archives/arch_logs.tab**, enter:

   ```
   splm -a archive -c -d /var/adm/archives -t /etc/tables/archive.tab
   ```

2. To create a **service** collection of entries in the /**spdata/sys1/logtables/amd.tab** table and have **snap** include general system information, enter:

   ```
   splm -a service -c -t /spdata/sys1/logtables/amd.tab -p g
   ```

3. To **gather** the **service** collections in Example 2, remove the collection on each node, and copy the gathered data to tape device rmt0, enter:

   ```
   splm -a gather -k service -t /spdata/sys1/logtables/amd.tab \
   -l /tmp/amdproblem -o /dev/rmt0 -r
   ```

splogd daemon

Purpose

splogd – Reports error logging, writes state changes, and calls user exits.

Syntax

splogd [–**d**] [–**b**] [–**f** *file_name*]

Flags

–**d** Turns debugging on.

–**b** Starts the daemon in the background from the command line.

–**f** *file_name*
Names an input file to use to define what logging is to be done and what user exits should be called. The default file is **/spdata/sys1/spmon/hwevents**.

Operands

None.

Description

The SP logging daemon has the following functions:

error logging
Reports SP hardware errors to both the syslog and the AIX error log.

state change logging
Writes SP hardware state changes to a file.

user exits Calls a user exit when a state change occurs.

The **hwevents** file contains state change actions that are to be performed by the **splogd** logging daemon. The fields are:

frame Specifies a frame number (1–*n*) or * for all frames.

slot Specifies the following:

- A number from 0–17
- One of:
  - NODES_ONLY (addresses 1 through 16)
  - SWITCH (address 17)
  - FRAME (address 0)
  - * (all addresses)
  - NODES_AND_SWITCH (addresses 1–17)
  - FRAME_AND_NODES (addresses 0–16)
  - FRAME_AND_SWITCH (addresses 0 and 17)

variable Specifies a hardware variable (for example, nodePower, temp, LED7SegA).

operator Specifies how to compare the value. Acceptable values are: **=**, <, >, and **!=**.

value Specifies the value of the variable to match with the operator wildcard (*), or a partial match with the wildcard at the end (23*).

time Specifies if the function should be called at startup, when the state changes, or both times. Valid options are **startup**, **change**, or **both**.

function Specifies the program to call when an event occurs.

There are two special keywords for function. If function is SP_ERROR_LOG, error logging is performed provided that syslog is set up and AIX error logging is set up to perform SP logging. Refer to the **setup_logd** command for details.

If function is SP_STATE_LOG, these state changes that meet the statement's criteria are logged to **/var/adm/SPlogs/spmon/splogd.state_changes.***timestamp*.

Note: To close the current **state_changes.***timestamp* and open a new one, send a SIGHUP signal to **splogd**. For example,

```
kill -HUP {splogd pid}
```

User Exit Arguments

When a user exit is called by **splogd**, the following arguments are passed:

1. A **c** or **s** depending on whether this call is for a change of state or to provide the startup values for the variables being monitored.
2. For each variable being reported, the following arguments are passed:
   a. Frame number.
   b. Node number.
   c. Variable name. Refer to the "System Monitor Variables, Display Types, and Attributes Appendix" of *IBM Parallel System Support Programs for AIX: Administration Guide* for a list of variables.
   d. Value of the variables. Boolean variables are expressed as TRUE or FALSE, integers as decimal strings, and floating-point values as floating-point strings.

Starting and Stopping the splogd Daemon

The **splogd** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **splogd** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **splogd**. In order to start the **splogd** daemon, use the **startsrc** –**s splogd** command. This starts the daemon with the default arguments and SRC options. The **splogd** daemon is setup to be respawnable and be the only instance of the **splogd** daemon running on a particular node or control workstation. Do **not** start the **splogd** daemon from the command line without using the **startsrc** command to start it.

To stop the **splogd** daemon, use the **stopsrc** –**s splogd** command. This stops the daemon and does not allow it to respawn.

To display the status of the **splogd** daemon, use the **lssrc** –**s splogd** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4*

Commands Reference and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget –q "subsysname=splogd" SRCsubsys** command.

Files

/spdata/sys1/spmon/hwevents
File that describes what logging is performed and what user exits are called.

/var/adm/SPlogs/spmon/splogd.state_changes.*timestamp*
File where state changes are recorded.

Related Information

Command: **setup_logd**

The "System Monitor Variables, Display Types, and Attributes Appendix" in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

1. To start the **splogd** daemon, enter:

   ```
   startsrc -s splogd
   ```

2. To stop the **splogd** daemon, enter:

   ```
   stopsrc -s splogd
   ```

3. To display the status of the **splogd** daemon, enter:

   ```
   lssrc -s splogd
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **splogd** daemon, enter:

   ```
   odmget -q "subsysname=splogd" SRCsubsys
   ```

splst_syspars

Purpose

splst_syspars – Returns the list of defined system partitions.

Syntax

splst_syspars [–**n**]

Flags

–**n** Returns a list of host names instead of addresses.

Operands

None.

Description

This command returns the list of the system partitions. The system partition names are in dotted decimal format unless –**n** is specified.

Examples

1. To display the IP addresses associated with all the defined system partitions on the SP, enter:

   ```
   splst_syspars
   ```

   You should receive output similar to the following:

   ```
   129.40.127.122
   129.40.127.47
   ```

2. To display the names of all the defined system partitions on the SP, enter:

   ```
   splst_syspars -n
   ```

   You should receive output similar to the following:

   ```
   k47sp1
   k47s
   ```

splst_versions

Purpose

splst_versions – Returns information about the PSSP code version installed on nodes in the SP system.

Syntax

splst_versions [–**G**] [–**l**] [–**e**] [–**n** *node_num*]
[–**N** *node_group*] [–**t**] [–**h**]

Flags

–**G** Causes the command to look at all system partitions rather than just the current system partition (but not the control workstation).

–**l** Returns the latest PSSP version for the nodes that are the target of the command.

–**e** Returns the earliest PSSP version for the nodes that are the target of the command.

–**n** *node_num*
Returns the PSSP code version for *node_num*. Use *node_num* 0 to specify the control workstation.

–**N** *node_group*
Returns a list of PSSP versions for *node_group*. If –**G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

–**t** Returns the node number and PSSP version in two columns.

–**h** Displays usage information.

Operands

None.

Description

Use this command to return a list of PSSP code versions that are installed on the nodes in the current system partition. The PSSP version and release numbers are included in the output. The modification level and fix level are not returned as part of the output. Node number 0 (zero) is considered the control workstation and is **not** evaluated as part of any system partition. The output is sorted in ascending order by version.

If the –**t** flag is omitted, there will be only one record for each version present. If the –**t** flag is used, there will be a record for each node.

Examples

1. To list each PSSP version represented in the current system partition, enter:

```
prompt> splst_versions
PSSP-1.2
PSSP-2.2
```

2. To list each node in the system partition and its PSSP code version, enter:

```
prompt> splst_versions -t
1 PSSP-2.1
5 PSSP-2.1
6 PSSP-2.1
9 PSSP-2.2
```

3. To list the earliest and latest PSSP code versions in a system partition, enter:

```
prompt> splst_versions -l -e
PSSP-2.1                    /* this case has mixed partitions */
PSSP-2.2
```

   The following will be the output if only PSSP-2.2 exists in the system partition:

```
prompt> splst_versions -l -e
PSSP-2.2                    /* this case has only 2.2 in partition */
```

splstdata

Purpose

splstdata – Displays configuration data from the System Data Repository (SDR) or system information for each node.

Syntax

splstdata {–**A** | –**n** | –**s** | –**b** | –**a** | –**u** | –**h** | –**i** | –**d**} [–**G**] [{*start_frame start_slot* {*node_count* | **rest**} | –**N** *node_group* | –**l** *node_list*}]

OR

splstdata {–**e** | –**f** | –**p**}

Flags

One of the following flags must be specified with each invocation of **splstdata**:

–**A** Displays the following SDR accounting data:

node_number
host_name
acct_class_id
acct_enable
acct_excluse_enable
acct_job_charge

–**n** Displays the following SDR node data:

node_number
frame_number
slot_number
slots_used
host_name
rel_host
default_route
processor_type
processors_installed

–**s** Displays the following SDR node switch data:

node_number
host_name
switch_node_number
switch_protocol
switch_number
switch_chip_number
switch_port_number

–**b** Displays the following SDR boot/install data.

node_number
host_name
hdw.enet.addr
boot_server
bootp_response

install_disk
next_inst_image
last_inst_image
last_inst_time

–a Displays the following SDR LAN data only for nodes in the current system partition:

node_number
adapter_type
netaddr
netmask
host_name
type
rate

–u Displays the following SDR /**usr** data:

node_number
host_name
usr_server_ip
usr_gateway_id
usr_client_adapter
has_usr_clients

–h Displays hardware data for each node, as provided by the **lscfg** command.

–i Displays network adapter data for each node, as provided by the **netstat** **–n** command.

–d Displays file system data for each node, as provided by the **df** command.

The following flags are optional:

–G Allows the specification of nodes to include one or more nodes outside of the current system partition.

–N *node_group*
Specifies a node group to be used for this operation. If **–G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

–l *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

–e Displays SP object attributes and their values from the SDR.

–f Displays the following SDR frame data:

frame_number
tty
frame_type

–p Lists all information for the currently-applied system partition configuration (all active system partitions on the system). This includes the list of system partitions, plus information about each system partition.

Operands

start_frame Frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive. If *start_frame*, *start_slot* and *node_count* are not specified, the default is **1 1 rest**. If *start_frame* is specified, *start_slot* and *node_count* must also be specified.

start_slot Slot number of first node to be used for this operation. Specify a value between 1 and 16 inclusive.

Note: *The start_frame* and *start_slot* must resolve to a node in the current system partition.

node_count Number of nodes to be used for this operation. Node information is provided for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive. If **rest** is specified, all the nodes from *start_frame start_slot* to the end of your system are used.

Note: The *node_count* is considered to be within the current system partition.

Description

You can use the System Management Interface Tool (SMIT) to run the **splstdata** command. To use SMIT, enter:

```
smit list_data
```

and select the System Data Repository option for the information you want to see. To see system information for each node, enter:

```
smit config_data
```

and select the option for the information you want.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

1. To display SDR node data for all the nodes in the SP system, enter:

   ```
   splstdata -n
   ```

   You should receive output similar to the following:

```
                    List Node Configuration Information

node frame slot
 #     #    #   slots initial_hostname  reliable_hostname default_route
-------------------------------------------------------------------------
 1     1    1     2   k47n01.hpssl.kgn  k47n01.hpssl.kgn  129.40.164.126
 3     1    3     2   k47n03.hpssl.kgn  k47n03.hpssl.kgn  129.40.164.126
 5     1    5     2   k47n05.hpssl.kgn  k47n05.hpssl.kgn  129.40.164.126
 7     1    7     2   k47n07.hpssl.kgn  k47n07.hpssl.kgn  129.40.164.126
 9     1    9     1   k47n09.hpssl.kgn  k47n09.hpssl.kgn  129.40.164.126
10     1   10     1   k47n10.hpssl.kgn  k47n10.hpssl.kgn  129.40.164.126
11     1   11     1   k47n11.hpssl.kgn  k47n11.hpssl.kgn  129.40.164.126
12     1   12     1   k47n12.hpssl.kgn  k47n12.hpssl.kgn  129.40.164.126
13     1   13     1   k47n13.hpssl.kgn  k47n13.hpssl.kgn  129.40.164.126
14     1   14     1   k47n14.hpssl.kgn  k47n14.hpssl.kgn  129.40.164.126
15     1   15     1   k47n15.hpssl.kgn  k47n15.hpssl.kgn  129.40.164.126
16     1   16     1   k47n16.hpssl.kgn  k47n16.hpssl.kgn  129.40.164.126
```

2. To display SDR boot/install data for the first four nodes in the first frame, enter:

   ```
   splstdata -b 1 1 4
   ```

   You should receive output similar to the following:

   ```
                    List Node Boot/Install Information
                       hdw_                          last_   last_   next_
   node                enet_                 inst_  inst_   inst_   inst_
    #  hostname         addr       srvr  resp  disk   image   time    image
   ------------------------------------------------------------------------
   1   k47n01.hpssl.kgn 02608C2D58D2 0 install hdisk0 initial initial deflt
   3   k47n03.hpssl.kgn 02608C2DA43C 0 install hdisk0 initial initial deflt
   5   k47n05.hpssl.kgn 02608C2DA37F 0 install hdisk0 initial initial deflt
   7   k47n07.hpssl.kgn 02608C2DA22A 0 install hdisk0 initial initial deflt
   ```

3. To list system partition information, enter:

   ```
   splstdata -p
   ```

   You should receive output similar to the following:

   ```
   System Partitions:
   ------------------
   k47sp1
   k47s

   Syspar: k47sp1
   -------------------------------------------------------------------------
   syspar_name     k47sp1
   ip_address      129.40.127.122
   install_image   bos.obj.ssp.41
   syspar_dir      /spdata/sys1/syspar_configs/1nsb0isb/config.8_8/
                   layout.3/syspar.1
   code_version    PSSP-2.1

   Syspar: k47s
   -------------------------------------------------------------------------
   syspar_name     k47s
   ip_address      129.40.127.47
   install_image   bos.obj.ssp.41
   syspar_dir      /spdata/sys1/syspar_configs/1nsb0isb/config.8_8/
                   layout.3/syspar.2
   code_version    PSSP-2.1
   ```

splsuser

Purpose

splsuser – Lists the attributes of an SP user account.

Syntax

splsuser [–**c** | –**f**] *name*

Flags

–**c** Displays the attributes for the user in colon-separated records.

–**f** Displays the attributes for the user in stanza format.

Operands

name Name of the user account you want to view.

Description

You can only list the information for one SP user at a time. Unlike the AIX **lsuser** command, the **ALL** option and the –**a** flag are not supported for this command.

If you specify this command with no flags, the information of the user appears in a sequential display of attribute and values.

You can use the System Management Interface Tool (SMIT) to run the **splsuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Change/Show Characteristics of a User option.

Examples

1. To display the attributes of the user account **rob** in a colon-separated list, enter:

   ```
   splsuser -c rob
   ```

   You should receive output similar to the following:

   ```
   #name:id:pgrp:groups:home:shell:gecos:login
   rob:16416:1::/u/rob on k46s.hpssl.kgn.ibm.com:/bin/ksh::true
   ```

2. To display the attributes of the user account **rob** in stanza format, enter:

   ```
   splsuser -f rob
   ```

   You should receive output similar to the following:

   ```
   rob:
           id=16416
           pgrp=1
           groups=
           home=/u/rob on k46s.hpssl.kgn.ibm.com
           shell=/bin/ksh
           gecos=
           login=true
   ```

spmkuser

Purpose

spmkuser – Adds a new user account to the SP system.

Syntax

spmkuser [*attribute*=*value* ...] *name*

Flags

None.

Operands

attribute=*value*	Pairs of the supported attributes and values as follows.
name	User login name. This name must follow the same rules enforced by the AIX **mkuser** command.

Supported Attributes and Values

id	ID of the user specified by the *name* operand.
pgrp	Principle group of the user specified by the *name* operand.
gecos	General information about the user.
groups	The secondary groups to which the user specified by the *name* operand belongs.
home	Host name of the file server where the home directory resides and the full path name of the directory. You can: • Specify a host and directory in the format *host:path*, just specify the directory and have the host default to a value set in the SMIT site environment panel or the **spsitenv** command. • Specify a directory and have the host default to the local machine.
login	Indicates whether the user specified by the *name* operand can log in to the system with the **login** command. This option does not change the /**etc**/**security**/**user** file. Instead, it alters the user password field in /**etc**/**security**/**passwd**.
shell	Program run for the user specified by the *name* operand at the session initiation.

Description

The –**a** flag is not supported. Except for **home**, the rules for the supported attributes and values correspond to those enforced by the AIX **mkuser** command.

All other attribute and value pairs are not supported.

The standard administrative AIX privileges do not apply to the SP users.

This command generates a random password for the user and stores it in **/usr/lpp/ssp/config/admin/newpass.log**. The root user has read and write permission to this file. It is the administrators responsibility to communicate this password to the new user and periodically delete the contents of this file.

You can use the System Management Interface Tool (SMIT) to run the **spmkuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Add a User option.

Note: The home directory must be in an exported file system before you can run this command.

Examples

Note

The following examples assume that Amd is running and the following defaults are specified:

spsitenv command or SMIT panel HOMEDIR_SERVER="svr1"
HOMEDIR_PATH="/home/filesvr1"

spmkuser.default file In the *user* stanza:

group=staff
groups=staff
prog=/bin/ksh

To create a user account for **baker** using the defaults specified in the **spmkuser.default** file and the home directory specified in the SMIT site environment panel or **spsitenv** command:

```
spmkuser baker
```

To create a user account for **charlie** with a UID of 1234, a home directory of **/u/charlie** that is physically located at **/home/charlie** on **hostx**, the **staff** primary group and the **dev**, the **test** secondary groups, and the **/bin/ksh** default shell:

```
spmkuser id=1234 groups=dev,test home=hostx:/home/charlie
```

spmon

Purpose

spmon – Operates the system controls and monitors system activity.

Syntax

spmon [–**query** [–**Monitor**] [–**long**] | –**connect** *host_name* | –**gui** | –**Global** | –**help** | –**key** {**normal** | **secure** | **service**} | –**Key** | –**Led** | –**power** {**on** | **off**} | –**reset** | –**mux** {**i** | **1** | **2** |**3**} | –**open** | –**diagnostics**] [[–**target**] *target_value*...]

Flags

All **spmon** commands require a –**target** parameter except those with –**diagnostics**, –**gui**, and –**help** parameters.

–**query** Queries the hardware variable specified as the target and returns the requested value. **query** is the default. If no other parameter is entered, a query is performed.

–**Monitor** Monitors the variables specified in the targets. If any of the specified variables change the state, the new state is written to standard output.

–**long** Applies only to –**query** and –**Monitor**. Returns the requested variables in fully qualified hierarchical format rather than the default format which is just the variable value.

–**connect** Connects to control workstation specified in *host_name* variable. Use this parameter with the –**key**, –**Key**, –**Led**, –**open**, –**power**, –**reset**, and –**mux** parameters.

–**gui** Starts the SP System Monitor graphical user interface.

–**Global** Allows targets that are outside of the current system partition. This parameter must be used for any query or command if specifying frames or switches.

–**help** Displays the usage information for the **spmon** command.

–**key** Choice of **normal** | **secure** | **service**. Changes the key mode switch position for the node specified as the target.

–**Key** Returns status of the key mode switch position for the node specified as the target.

–**Led** Displays the 3-digit display value.

–**power** Choice of **on** | **off**.

Turns the power on or off for the node, frame, or switch specified as the target. For example:

```
spmon -G -p off frame1
spmon -p off node16
spmon -G -p off frame2/switch
spmon -G -p off frame11/switch2
```

–reset Resets the node specified as the target.

–mux Choice of **i** | **1** | **2** | **3**. Sets multiplexors that control the clocking of a switch to the value indicated. These values mean:

i Use internal oscillator (make this switch the master)
1 Use input 1
2 Use input 2
3 Use input 3

The **mux** setting must match the physical wiring of the switch clocks and requires a frame as its target. For a switch in node 17, use a frame as the target or frame/switch*N* for a switch in a switch-only frame.

–open Opens a **tty** connection to the node specified in the target flag. Press *Enter* to begin the session. Type **Ctrl-x** to close the connection. Refer to the **s1term** command for details.

–diagnostics Performs the following diagnostics tests:

1. Checks if the server process is running
2. Tries to open a connection to the server
3. Queries the number of frames in system
4. If the **–G** parameter is specified, for each frame checks:
   - If the frame controller responding
   - If a High Performance Switch is attached
   - The **mux** value
   - If the frame power supplies are ON or OFF
5. For each node in each frame, checks:
   - Node type
   - If power is on or off
   - hostResponds
   - switchResponds
   - The position of key switch
   - Environment failure
   - The values of the front panel LEDs.

   For each switch, checks:
   - Frame, slot
   - Node type
   - If power is on or off
   - Clock input
   - Environment failure

The tests are in dependent order. If any of these fail, the subsequent tests do not run.

[–target] *target_value*

Specifies the target node, frame, variable, or attribute for the command as *target_value*.

The **–target** flag is optional. Any parameter without a flag is assumed to be the target. You can also have multiple target-flags (**–t**), which are optional.

Operands

None.

Description

Any unique abbreviation of flags and keywords is acceptable.

Specify *target_value* with the hierarchical format (or tree structure). The format is:

`/SP/frame/frameN/[nodeM|switchM]/variableX/value`

SP	Is literally the string “SP”.
`frame`	Is the string **frame**.
`frameN`	Is frame1...frame*N* where *N* is the frame number in the SP system.
`nodeM` | `switchM`	Is the node number or switch number within the specified frame. *M* is the slot number of that node or switch. When switch is specified without a number, it means switch 17.
variableX	Is a variable known to the SP System Monitor. Refer to the “System Monitor Variables, Display Types, and Attributes Appendix” of *IBM Parallel System Support Programs for AIX: Administration Guide* for a list of variables.
value	Is literally the string “value.”

You can use wildcards (*) to specify more than one target node or frame for the **query** command.

Note: Though they are not hardware variables, for compatibility with older systems, the variables *hostResponds* and *switchResponds* can be used as specific targets of the **spmon** command for both –**query** and –**Monitor** commands. However, the variable names must be entered explicitly. These two variables are not returned if the variable specified is a wildcard (*).

You can use aliases in place of fully qualified hierarchical target values. Aliases require less typing and may be more intuitive than the fully qualified targets. Leaving the leading slash (/) off the target indicates that it is an alias.

There are two formats for aliases:

- Format 1: `frameN/nodeM`

`frameN`	Is the target frame, where *N* is the frame number.
`nodeM`	Is node1 to node16 within the specified frame.

  You can include a variable and attribute after the alias.

- Format 2: `nodeM`

`nodeM`	Is node1 to node*M* where *M* is the node number of the node in the SDR node class.

Examples

1. To query the key setting of node1 on frame1, enter:

   ```
   spmon -q -t /SP/frame/frame1/node1/keyModeSwitch/value
   0
   ```

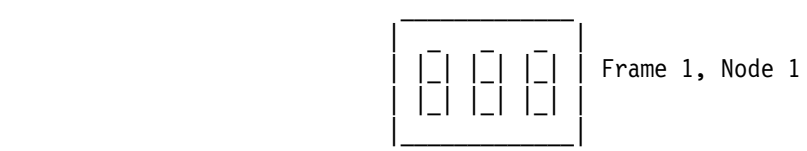
2. To perform the same query using an alias (uses query flag default), enter:

   ```
   spmon node1/keyModeSwitch/value
   0
   ```

3. To query the LED settings of node1 on frame1, enter:

   ```
   spmon -L frame1/node1
   ```

   You should receive output similar to the following:

4. To query the **mux** value on all switches in the system, enter:

   ```
   spmon -G -q -l frame1/switch*/mux/value
   /SP/frame/frame1/switch17/mux/value/0
   ```

5. To monitor the power LEDs on the nodes on frame1, enter:

   ```
   spmon -M frame1/node*/powerLED/value
   2
   1
   ```

6. To query the power LEDs on the nodes on frame1 and then monitor them and print the values in fully qualified hierarchical form, enter:

   ```
   spmon -M -q -l frame1/node*/powerLED/value
   /SP/frame/frame1/node1/powerLED/value/1
   /SP/frame/frame1/node3/powerLED/value/1
   /SP/frame/frame1/node5/powerLED/value/1
   /SP/frame/frame1/node7/powerLED/value/1
   /SP/frame/frame1/node9/powerLED/value/1
   /SP/frame/frame1/node10/powerLED/value/1
   /SP/frame/frame1/node11/powerLED/value/1
   /SP/frame/frame1/node12/powerLED/value/1
   /SP/frame/frame1/node13/powerLED/value/1
   /SP/frame/frame1/node14/powerLED/value/1
   /SP/frame/frame1/node15/powerLED/value/1
   /SP/frame/frame1/node16/powerLED/value/1
   /SP/frame/frame1/node1/powerLED/value/2
   /SP/frame/frame1/node1/powerLED/value/1
   ```

   Note: "node*" returns powerLED values on switches in slots 1—16 also.

7. To power off node3 on frame2, enter:

   ```
   spmon -p off frame2/node3
   ```

   If node3 on frame2 is outside the current system partition, enter:

   ```
   spmon -G -p off frame2/node3
   ```

8. To power off node3 on frame2 using alias format2, enter:

   ```
   spmon -p off node19
   ```

9. To change key setting on node1 on frame1 to service, enter:

   ```
   spmon -k service node1
   ```

10. To power off frame1, (type 17 frame supervisor only), enter:

    ```
    spmon -G -p off frame1
    ```

11. To power off frame1, (SEPBU - type 18 frame supervisor), enter:

    ```
    spmon -G -p off frame1/A
    ```

12. To set the frame1 switch to be the master switch (use internal oscillator), enter:

    ```
    spmon -G -m i frame1
    ```

    or

    ```
    spmon -G -m i frame1/switch
    ```

13. To set frame 10, switch4 in a switch-only frame to be the master switch, enter:

    ```
    spmon -G -m i frame10/switch4
    ```

spmon_ctest

Purpose

spmon_ctest – Verifies that the System Monitor component is configured correctly.

Syntax

spmon_ctest [–**l** *log_file*] [–**q**]

Flags

–**l** *log_file* Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

–**q** Specifies quiet mode; suppresses output to standard error.

Operands

None.

Description

This command is designed to be run after installing the SP system to verify that the System Monitor is configured correctly. The test checks to make sure that the hardware is running, that it can be queried, and determines whether any node objects were created in the System Data Repository (SDR). The test also indicates whether the RS232 lines are connected properly.

A return code of zero indicates that the test completed as expected; otherwise it returns the number of failures. If you do not specify the –**q** flag, a message is displayed on standard output that indicates the success or failure of the tests. In either case, the command returns 0 if successful, 1 if not. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in /**var**/**adm**/**SPlogs**/**spmon_ctest.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Monitor Configuration option.

You must run this test from a user who has monitor authority in /**spdata**/**sys1**/**spmon**/**hmacls**. The user must also have a nonexpired authentication ticket.

Refer to Chapter 2, “SP Files and Other Technical Information” section of *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for additional **Kerberos** information.

Files

/usr/lpp/ssp/bin/spmon_ctest
Path name of this command.

/var/adm/SPlogs/spmon_ctest.log
Default log file.

Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_itest**

Examples

To verify installation of the SP System Monitor, saving error messages in **spmon.err** in the current working directory, enter:

```
spmon_ctest -l spmon.err
```

spmon_itest

Purpose

spmon_itest – Verifies that the System Monitor is installed and operational.

Syntax

spmon_itest [–**l** *log_file*] [–**q**]

Flags

–**l** *log_file* Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

–**q** Specifies quiet mode; suppresses output to standard error.

Operands

None.

Description

This command is designed to be run after installing the SP system to verify that the System Monitor is installed correctly.

A return code of zero indicates that the test completed as expected; otherwise it returns the number of failures. If you do not specify the –**q** flag, a message is displayed on standard output that indicates the success or failure of the tests. In either case, the command returns 0 if successful, 1 if not. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/spmon_itest.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Monitor Installation option.

Files

/usr/lpp/ssp/bin/spmon_itest
Path name of this command.

/var/adm/SPlogs/spmon_itest.log
Default log file.

Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_ctest**

Examples

To verify installation of the SP System Monitor, saving error messages in **spmon.err** in the current working directory, enter:

```
spmon_itest -l spmon.err
```

spperfmon

Purpose

spperfmon – Directly launches the Performance Monitor window of the SP Perspectives graphical user interface (GUI).

Syntax

spperfmon [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[{–**backgroundColor** | **bg**} *colorName*]
[{–**foregroundColor** | **fg**} *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to launch the SP Performance Monitor window of the SP Perspectives GUI. This tool enables the user to monitor the performance of the SP in conjunction with other licensed products: Performance Toolbox for AIX (PTX), 5765-654 and Performance Toolbox Parallel Extensions for AIX (PTPE), 5765-529.

From the Performance Monitor window, you can perform most of the PTPE command set functions through point and click operations. For example, you can easily manipulate the PTPE monitoring hierarchy and save it to the System Data Repository (SDR).

The Performance Monitor perspective window uses three panes to display SP system information:

1. *Hierarchy pane*: This shows the current monitoring hierarchy, displaying the central coordinator at the top, with data manager nodes below and reporter nodes at the bottom. By default, the monitoring hierarchy from the System Data Repository (SDR) is displayed when this perspective is initialized.
2. *Syspar pane*: This shows how the SP is partitioned. The system partition selected in this pane is the one displayed in the Hierarchy and Nodes panes. If other partitions are defined by the SP, you can use this pane to select them.
3. *Nodes pane*: This shows the nodes in the SP system, organized by frame in the default display, but you can sort and filter them to suit your purposes.

When the command is invoked, preferences that define the look and layout of the SP Performance window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

- **/usr/lpp/ssp/bin/spperfmon** is the path name of this command.
- The Users Preferences are read from and saved to **$HOME/.spperfmon(User Profile Name)**.
- The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spperfmon(System Profile name)**.

Restrictions

Any user can run the **spperfmon** command. To get a read/write PTPE session requires root privilege and the user must be a member of the UNIX group 'perfmon'.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP) and the IBM Performance Toolbox Parallel Extensions for AIX separately priced feature.

Prerequisite Information

For information on using **spperfmon** and the SP Perspectives GUI, please see the online help and the "Using the SP Perspectives GUI" chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*.

Location

/usr/lpp/ssp/bin/spperfmon

Related Information

You can also access the Performance Monitor window by using the SP Perspectives Launch Pad. The **perspectives** command invokes the launch pad. Other Perspectives windows can be launched by invoking the following commands: **spevent**, **sphardware**, **spsyspar**, and **spvsd**.

IBM Performance Toolbox Parallel Extensions for AIX: Guide and Reference

IBM Performance Toolbox 1.2 and 2.1 for AIX: Guide and Reference

Examples

1. To invoke the spperfmon window, enter:

   ```
   spperfmon
   ```

2. To launch the SP Performance Monitor Perspective ignoring the preferences found in the system and user profile files, enter:

   ```
   spperfmon -noProfile
   ```

sprestore_config

Purpose

sprestore_config – Restores the system to a given system partitioning configuration as specified in the System Data Repository (SDR) which was previously archived.

Syntax

sprestore_config *archive_file* [–**h**]

Flags

–**h** Displays usage information.

Operands

archive_file Specifies the name of the archived SDR file to be restored.

Description

Use this command to restore the SDR from an archive file that was previously created with the **SDRArchive** command. In addition to restoring the SDR (using the **SDRRestore** command), the **sprestore_config** command also restores system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) to their previous state. This command is most useful when recovering from an attempt to partition the SP (see the **spapply_config** command).

You can use the System Management Interface Tool (SMIT) to run the **sprestore_config** command. To use SMIT, enter:

```
smit syspar_restore
```

and enter (or select from a generated list) the name of the SDR archive from which to restore.

Notes:

1. This command should be run only on the control workstation.
2. Due to system partitioning changes, your SP_NAME environment variable may no longer be set to a valid system partition name. To get a list of valid system partition names, enter the **splst_syspars -n** command. Then verify that your SP_NAME environment variable is either unset or set to one of the system partition names in the list.

Exit Values

0 Indicates success.

1 Indicates that an error occurred while trying to restore the specified system partitioning configuration.

2 Indicates a usage error.

Related Information

Commands: **SDRArchive**, **SDRRestore**, **spapply_config**, **spcustomize_syspar**, **spdisplay_syspar**, **spverify_config**, **syspar_ctrl**

Files: **nodelist**, **topology**

Examples

To restore the SDR and the system-partition sensitive subsystems (for example, **hats**, **hb**, and **hr**) from the archive **"backup.95110.1620"** which was previously created using the **SDRArchive** command, enter:

```
sprestore_config backup.95110.1620
```

sprmuser

Purpose

sprmuser – Removes a user account from the SP system.

Syntax

sprmuser [–**i**] [–**p**] [–**r**] *name*

Flags

–**i** Displays the current user information and enables interactive control. This allows you to quit before deleting the user account.

–**p** Removes user password information from the /**etc**/**security**/**passwd** file.

–**r** Removes the user's home directory specified in the **home** attribute.

name Name of the user account you want to delete.

Operands

None.

Description

The –**i** and –**r** options are unique to the SP system.

You can use the System Management Interface Tool (SMIT) to run the **sprmuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Remove a User option.

Examples

1. To remove user account **charlie** without destroying the home directory, enter:

   ```
   sprmuser charlie
   ```

2. To remove user account **charlie**, remove any information about this user in the /**etc**/**security**/**passwd** file, and remove the home directory, enter:

   ```
   sprmuser -pr charlie
   ```

spsitenv

Purpose

spsitenv – Enters configuration parameters used by SP installation and system management scripts into the System Data Repository (SDR).

Syntax

spsitenv [**acct_master** = *accounting_master*]
[**amd_config** = **true** | **false**]
[**cw_lppsource_name** = *lppsource_name*]
[**filecoll_config** = **true** | **false**]
[**homedir_path** = *home_directory_path*]
[**homedir_server** = *home_directory_server_host_name*]
[**install_image** = *default_network_install_image_name*]
[**ntp_config** = **consensus** | **internet** | **none** | **timemaster**]
[**ntp_server** = *ntp_server_host_name* ...]
[**ntp_version** = **3** | **1**]
[**passwd_file_loc** = *passwd_file_server_host_name*]
[**passwd_file** = *passwd_file_path*]
[**print_config** = **false** | **open** | **secure**]
[**print_id** = *secure_print_login_name*]
[**remove_image** = **true** | **false**]
[**spacct_enable** = **false** | **true**]
[**spacct_actnode_thresh** = *sp_accounting_active_node_threshold*]
[**spacct_excluse_enable** = **false** | **true**]
[**supfilesrv_port** = *port*]
[**supman_uid** = *supman_uid*]
[**usermgmt_config** = **false** | **true**]

Flags

acct_master Indicates which node is the accounting master, where **crunacct** runs. The initial value is *accounting_master*=0, specifying the control workstation.

amd_config Indicates whether Amd should be configured on your SP. Specify **true** if you want to have AMD configured and to have Amd entries created for your home directories. Specify **false** if you do not want to have Amd started. The initial value of **amd_config** is **false**.

cw_lppsource_name
Indicates the LPP source name to use when installing the NIM file sets on the control workstation. The name you specify

must correspond to an LPP source directory on the control workstation. The directory must be named /**spdata**/**sys1**/**install**/*LPP_source_name*/**lppsource**, where *LPP_source_name* is the name that you have assigned to the LPP source. The default value is **default**.

filecoll_config Indicates whether the SP file collection management code should be installed.

Specify **true** if the code is to be installed. Specify **false** if the code is not to be installed. The initial value is **true**.

homedir_path Specify an absolute path name for *home_directory_path* if you want to use a path other than the initial value of the control workstation path. The initial value is /**home**/*cw*, where *cw* is the host name of the control workstation.

homedir_server Indicates where user home directories physically reside. Specify a valid host name or IP address for *home_directory_server_host_name* if you want to use a server other than the initial value of the control workstation host name.

The initial value is the host name of the control workstation.

install_image Indicates the location of the default network install image for your SP system. *default_network_install_image_name* should point to the install image which is used for a node if that node's install image field is not set. This should be a file in /**spdata**/**sys1**/**install**/**images**. The initial value is **bos.obj.ssp.41**.

ntp_config See **ntp_server**.

ntp_server Indicates your choice for running NTP in your SP. To use your site's NTP time server, specify **ntp_config=timemaster** and specify the host name of your NTP time server with the **ntp_server** parameter.

To use an Internet NTP time server, your control workstation must be connected to the Internet. Specify **ntp_config=internet** and specify the full host name of an Internet time server with the **ntp_server** parameter.

To cause the control workstation and file servers to generate a consensus time based on their own date settings, specify **ntp_config=consensus** and specify **ntp_server=""**.

If you do not want to run NTP on the SP, specify **ntp_config=none** and **ntp_server=""**.

The initial value of **ntp_config** is **consensus** and the initial value of **ntp_server** is **""**. If **ntp_config** is specified as either **timemaster** or **internet**, the **ntp_server** value must be a valid host name.

ntp_version Indicates which version of NTP you are running. The initial value is **3**.

passwd_file Indicates the path of the password file where new user entries are placed. The initial value is /**etc**/**passwd**. If you change the value of **passwd_file** from /**etc**/**passwd** and are using NIS, be

sure to modify your NIS Makefile to build the password map from the new password file.

This field is meaningful only if **usermgmt_config=true**.

passwd_file_loc Specifies the host name of the machine where your password file resides. The initial value of **passwd_file_loc** is the control workstation. The value of the **passwd_file_loc** cannot be one of the nodes in the SP system.

print_config Indicates how SP print management should be integrated into your system.

Specify **print_config = false** if you do not want to have the existing AIX print commands saved and have the file names linked to the SP print functions.

If you want to have the print management system integrated, specify **secure** if you do not want users to not have rsh privileges on the print host. Specify **open** if the users are to have rsh privileges on the print host.

The initial value of **print_config** is **false**.

print_id Specify the login name to be used for rsh for secure mode printing. This field is meaningful only if **print_config=secure** is specified.

The initial value is **""**. If the value remains at **""** and the value of **print_config** is **secure**, the installation script uses a print management ID of **prtid** as a default.

remove_image Indicates whether install images are to be removed from the boot servers after an install has been completed.

Specify **remove_image=true** if the images are to be removed.

Specify **remove_image=false** if the images are not to be removed.

The initial value is **false**.

spacct_actnode_thresh
Indicates the percentage of nodes for which accounting data must be present in order for **crunacct** to continue processing that day. The initial value is **80**.

spacct_enable Indicates whether accounting is enabled or disabled on all nodes that have an accounting enabled attribute set to default. The initial value is **false**, disabling accounting.

spacct_excluse_enable
Indicates if accounting start and end job records are generated for jobs having exclusive use of the node. A value of **true** indicates that exclusive use accounting is enabled and start and end job records are generated. A value of **false** indicates that exclusive use accounting is not enabled and start and end job records are not generated.

The initial value is **false**.

supfilesrv_port Specifies the file collection daemon port. This is used in /**etc**/**services** for the file collection daemon. Pick a value that does not conflict with any other ports in use. It is meaningful only if **filecoll_config=true** is specified. The initial value is **8431**.

supman_uid Specifies the uid for the file collection daemon. It is meaningful only if **filecoll_config=true** is specified. The initial value is **""**. If you are using login control, make this uid lower than the threshold ID you set in the **block_usr_sample** script.

usermgmt_config Indicates whether SP user management scripts should be integrated into your system.

Specify **usermgmt=true** if you want to have the SP User Management scripts in the Security & Users SMIT menu. Specify **usermgmt=false** to remove the scripts from the SMIT menu.

The initial value is **false**.

Operands

None.

Description

Use this command during installation of the SP or at a later time to identify SP configuration parameters in use at your location.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **kinit**.

You can use the System Management Interface Tool (SMIT) to run the **spsitenv** command. To use SMIT, enter:

```
smit enter_data
```

and select the Site Environment Information option.

You cannot use SMIT if you are using AFS authentication services.

Note: This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

Examples

The following example enters site environment parameters into the System Data Repository. The NTP configuration is **consensus** and the file collection management code is to be installed:

```
spsitenv ntp_config=consensus filecoll_config=true
```

spsvrmgr

Purpose

spsvrmgr – SP Supervisor Manager.

Syntax

spsvrmgr [–**G**] [–**f** *file_name*]
[[–**q rc** | **msg**] |
[–**r status** | **action**] |
[–**m status** | **action**] | [–**u**]] [*slot_spec* | **all**]

Flags

–**G** Specifies Global mode. With this flag, commands can be sent to any hardware.

–**f** Uses *file_name* as the source of slot ID specifications.

–**q rc | msg** Checks the supervisor hardware configuration for supervisors that support microcode download, and that also require an action.

Action checks include:

Install Indicates that the Supervisor card has no supervisor installed. An install is required.

Upgrade Indicates that the Supervisor card has a supervisor installed, but it is not at the most current level. An upgrade is required.

Reboot Indicates that the Supervisor card has a supervisor installed and it is at the most current level, but it is not active. A reboot is required.

Update Media
Indicates that the Supervisor card has a supervisor installed, but the media that is the repository for microcode files does not contain the version that is installed on the card. A media update is required.

If **rc** is specified with the –**q** flag, the command will issue a return code indicating whether any of the hardware requires action. A return code of 0 indicates that no action is required. A return code of 1 indicates that at least one supervisor was found that required action.

If **msg** is specified with the –**q** flag, the command will issue a message indicating whether any of the hardware requires action. In this case, a return code of 0 is issued unless an error condition occurs.

–**r status | action**
Checks the supervisor hardware configuration for supervisors that support microcode download and displays status for those supervisors in “report” form.

If **status** is specified with the –**r** flag, the status is listed for all of the installed supervisors that support microcode download.

If **action** is specified with the –**r** flag, the status is listed for all of the installed supervisors that support microcode download and that also require an action.

In both cases, Status includes:

Frame Number
Indicates the number of the frame.

Slot Number Indicates a number in the range of 0—17.

Supervisor State
Indicates either Active (supervisor is executing) or Inactive (supervisor is not executing).

Media Versions
Indicates the microcode files that are compatible with the supervisor installed in this frame/slot.

Installed Version
Indicates the microcode file installed as the supervisor.

Required Action
Can be one of the following: None, Install, Upgrade, Reboot, or Update Media.

–**m status | action**
Checks the supervisor hardware configuration for supervisors that support microcode download and displays status for those supervisors in "matrix" form.

If **status** is specified with the –**m** flag, the status is listed for all of the installed supervisors that support microcode download.

If **action** is specified with the –**m** flag, the status is listed for all of the installed supervisors that support microcode download and that also require an action.

In both cases, Status includes:

Frame Number
Indicates the number of the frame.

Slot Number Indicates a number in the range of 0—17.

Action Required
Can be either Required or Not Required.

–**u** Installs, upgrades, or reboots the hardware supervisors specified by the *slot_spec* option that support microcode download and that also requires an action.

Note: This flag starts an **hmcmds** process to perform the actual update. Refer to the **hmcmds** command specifically the **basecode**, **microcode**, and the **boot_supervisor** command options.

Attention

In most cases, the –**u** flag started processes powers off the target slots during the duration of the update.

Operands

slot_spec | **all**
Specifies the addresses of the hardware components.

Description

The design of the SP supervisor control system divides the microcode used in the frame supervisor, node supervisor, and switch supervisor into the following two types:

basecode Microcode that is loaded at the time of manufacture and gives the card the ability to load application microcode during system operation.

application microcode Microcode that is loaded via basecode and contains the instruction that is the supervisor application.

The **spsvrmgr** command controls the software level and state of the supervisor applications that reside on the SP supervisor hardware.

Normally, commands are only sent to the hardware components in the current system partition. A system partition contains only processing nodes. The switches and the frames themselves are not contained in any system partition. To access hardware components not in the current system partition or to any frame or switch, use the –**G** flag.

The *slot_spec* option is interpreted as slot ID specifications. A slot ID specification names one or more slots in one or more SP frames and has either of two forms:

```
fidlist:sidlist   or   nodlist
```

where:

fidlist = fval[,fval,...]

sidlist = sval[,sval,...]

nodlist = nval[,nval,...]

The first form specifies frame numbers and slot numbers. The second form specifies node numbers. An *fval* is a frame number or a range of frame numbers of the form *a–b*. An *sval* is a slot number from the set 0 through 17 or a range of slot numbers of the form *a–b*. An *nval* is a node number or a range of node numbers of the form *a–b*.

The relationship of node numbers to frame and slot numbers is shown in the following formula:

$$node\_number = ((frame\_number - 1) \times 16) + slot\_number$$

Note: Node numbers can only be used to specify slots 1 through 16 of any frame.

Refer to the **hmcmds** command for examples of the *slot_spec*.

Optionally, slot ID specifications can be provided in a file rather than as command flags. The file must contain one specification per line. The command requires that slot ID specifications be provided. If the command is to be sent to all SP hardware, the keyword **all** must be provided in lieu of the *slot_spec* option. However, the **all** keyword can only be specified if the –**G** flag is specified.

Files

The media that is the repository for the application microcode files is the **/spdata/sys1/ucode** directory structure.

Exit Values

0 Indicates the successful completion of the command.

1 Returned only in conjunction with the **–q rc** flag to indicate that at least one supervisor was found that required action.

–1 Indicates that the command failed. This return value is always accompanied with an error message.

Restrictions

IBM recommends that you use this command through the RS/6000 SP Supervisor Manager option of the System Management Interface Tool (SMIT).

To access this command using SMIT, enter:

```
smit
```

and select the RS/6000 SP System Management option, then the RS/6000 SP Supervisor Manager option.

A list of options that correspond to the **spsvrmgr** command flags will be presented for selection.

You can also directly access this list of options using the following SMIT fast-path command:

```
smit supervisor
```

Implementation Specifics

You must be authorized to access the Hardware Monitor subsystem to run the **spsvrmgr** command. In addition, for those frames specified to the command, you must have Virtual Front Operator Panel (VFOP) permission. Commands sent to frames for which you do not have VFOP permission are ignored. Since the Hardware Monitor subsystem uses SP authentication services, you must run the **kinit** command prior to running this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

The **spsvrmgr** command, by design, only interacts with SP supervisor hardware that supports the ability to download application microcode. Commands sent to slots that do not support this ability are ignored.

Location

/usr/lpp/ssp/bin/spsvrmgr

Related Information

Commands: **hmcmds**

Refer to the “Installing and Configuring a New RS/6000 System” chapter of the *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*

Examples

1. To perform a “quick check” of your configuration for supervisor hardware that requires action and to have a message issued, enter:

   ```
   spsvrmgr -G -q msg all
   ```

   You should receive output similar to the following:

   ```
   spsvrmgr: At least one occurrence of supervisor hardware was found to
             require attention.
             Enter "smit supervisor" for installation options.
   ```

2. To perform a “quick check” of your configuration for supervisor hardware that requires action and to have a status code returned, enter:

   ```
   spsvrmgr -G -q rc all
   echo $?
   ```

   Example usage in a script:

   ```
   spsvrmgr -G -q rc all
   if [[ $? = 1 ]]
   then
       echo "*** Attention*** One or more supervisors require action."
       echo "Enter \"smit supervisor\" for installation options."
   fi
   ```

3. To display status information in report form of all hardware that supports microcode download for frame 2, enter:

   ```
   spsvrmgr -G -r status 2:0-17
   ```

   You should receive report output similar to the following:

   ```
   spsvrmgr: Frame Slot Supervisor Media        Installed    Required
                        State      Versions     Version      Action
             _____ ____ __________ ____________ ____________ ____________
             2     1    Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
                   ____ __________ ____________ ____________ ____________
                   5    Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
                   ____ __________ ____________ ____________ ____________
                   9    Active     u_10.1a.0609 u_10.1a.060b None
                                   u_10.1a.060a
                                   u_10.1a.060b
                   ____ __________ ____________ ____________ ____________
                   13   Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
   ```

4. To display status information in matrix form of all hardware that supports microcode download for in your configuration, enter:

```
spsvrmgr -G -r status all
```

You should receive matrix output similar to the following:

```
spsvrmgr: Frame       Slots
          _____       ____________________________________________
          1           00 01 05 09 13 17
             (Action)  -  -  -  -  -  -

          _____       ____________________________________________
          2           01 05 09 13
             (Action)  +  +  +  -

          Action Codes:
              +  -- Required
              -  -- Not Required
```

5. To display status information in report form of all hardware that supports microcode download and requires an action for frame 1, enter:

   ```
   spsvrmgr -G -r action 1:0-17
   ```

   You should receive report output similar to the following:

   ```
   spsvrmgr: Frame Slot Supervisor Media        Installed    Required
                        State      Versions     Version      Action

             _____ ____ __________ ____________ ____________ ____________
             1     1    Active     u_10.3a.0609 u_10.3a.060a Upgrade
                                   u_10.3a.060a
                                   u_10.3a.060b

                   ____ __________ ____________ ____________ ____________
                   5    Inactive   u_10.3a.0609 u_10.3a.060b Reboot
                                   u_10.3a.060a
                                   u_10.3a.060b

                   ____ __________ ____________ ____________ ____________
                   9    Inactive   u_10.1a.0609 u_10.1a.060b Reboot
                                   u_10.1a.060a
                                   u_10.1a.060b
   ```

6. To update the hardware that supports microcode download in frame 1 slot 1, enter:

   ```
   spsvrmgr -u 1:1
   ```

   You should receive installation output similar to the following:

   ```
   spsvrmgr: Dispatched "microcode" process [24831] for frame 1 slot 1.
             Process will take approximately 12 minutes to complete.

   spsvrmgr: Process [24831] for frame 1 slot 1 completed successfully.
   ```

7. To update the hardware that supports microcode download in frame 1 slots 5 and 9, enter:

   ```
   spsvrmgr -u 1:5,9
   ```

   You should receive installation output similar to the following:

```
spsvrmgr: Dispatched "boot_supervisor" process [27956]
          for frame 1 slot 5.
          Process will take less than a minute to complete.

spsvrmgr: Dispatched "boot_supervisor" process [23606]
          for frame 1 slot 9.
          Process will take less than a minute to complete.

spsvrmgr: Process [27956] for frame 1 slot 5 completed successfully.
spsvrmgr: Process [23606] for frame 1 slot 9 completed successfully.
```

spsyspar

Purpose

spsyspar – Directly invokes the System Partitioning Aid window of the SP Perspectives graphical user interface (GUI).

Syntax

spsyspar [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[{–**backgroundColor** | **bg**} *colorName*]
[{–**foregroundColor** | **fg**} *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to launch the System Partitioning Aid window of the SP Perspectives GUI. The System Partitioning Aid Perspective is used to view and manage the current system partitioning configuration. This tool can also be used to generate new configurations.

When the command is invoked, preferences which define the look and layout of the System Partitioning Aid window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

The users preferences are read from and saved to **$HOME/.spsyspar(User Profile Name)**. The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spsyspar(System Profile name)**. If a new system partitioning configuration is created, the following files are created under the layout directory: **layout.desc**, **nodes.syspar** and a system partition directory for each system partition in the layout. For each system partition directory, a node list file and topology file are created.

Restrictions

Any user can run the **spsyspar** command. Many actions require root privilege to run.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For information on using the System Partitioning Aid window and the SP Perspectives GUI, please see the online help and the “Using the SP Perspectives GUI” chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*.

Refer to the “Managing System Partitions” chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the System Partitioning Aid.

Location

/usr/lpp/ssp/bin/spsyspar

Related Information

You can also access the hardware window by using the SP Perspectives Launch Pad. The **perspectives** command invokes the launch pad. Other Perspectives windows may be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, and **spvsd**. The **sysparaid** command provides a command line interface into the System Partitioning Aid.

Examples

1. To launch the Partitioning Aid Perspective, enter:

   ```
   spsyspar
   ```

2. To launch the Partitioning Aid Perspective with a hot pink background regardless of what is provided in the preference files, enter:

   ```
   spsyspar -background hot_pink
   ```

spverify_config

Purpose

spverify_config – Verifies the active system partition configuration information for the SP system.

Syntax

spverify_config

Flags

None.

Operands

None.

Description

This command is run by the **spapply_config** command after the System Data Repository (SDR) is updated. It can also be run by an administrator to verify that the SDR information is consistent (such as, after a system outage or a problem with the SDR). (This verification is only performed on a system which was partitioned beyond the initial single partition created at initial installation.)

Exit Values

0 Indicates that the SDR and corresponding layout directory are in agreement.

1 Indicates differences were found.

2 Indicates a usage error.

Related Information

Commands: **spapply_config**, **spcustomize_syspar**, **spdisplay_config**

Files: **nodelist**, **topology**

Examples

To verify that the information in the SDR matches the customization information previously supplied by the user, enter:

```
spverify_config
```

spvsd

Purpose

spvsd – Directly launches the IBM Virtual Shared Disk window of the SP Perspectives graphical user interface (GUI).

Syntax

spvsd [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**] [{–**backgroundColor** | **bg**} *colorName*] [{–**foregroundColor** | –**fg**} *colorName*] [–**fontFamily** *name*] [–**fontSize** *size*] [–**fontBold**] [–**fontItalic**]

Flags

–**userProfile** *name*
Upon initialization, overrides the name of the user profile from the default value of 'Profile'.

–**systemProfile** *name*
Upon initialization, overrides the name of the system profile from the default value of 'Profile'.

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor | bg** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**foregroundColor | fg** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *IBM Parallel System Support Programs for AIX: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

Note: Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

Operands

None.

Description

Use this command to launch the IBM Virtual Shared Disk window of the SP Perspectives GUI. This perspective allows the user to view and control the IBM Virtual Shared Disk system.

By default, when the window is brought up, it displays:

- The system partitions
- The IBM Virtual Shared Disk nodes
- The IBM VSD/HSD definitions work areas

The current system partition will be selected within the system partitions. The IBM VSD Nodes pane displays all nodes in the current system partition. The IBM VSDs and HSDs Definitions pane displays all existing IBM VSD/HSD definitions within the system partition. You can control which panes are displayed by using the Add pane and Delete pane combo boxes.

When the command is invoked, preferences that define the look and layout of the **spvsd** window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

Files

The Users Preferences are read from and saved to **$HOME/.spvsd(User Profile Name)**. The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spvsd(System Profile name)**.

Restrictions

Any user can run the **spvsd** command. Many actions require root privilege to run.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Prerequisite Information

For information on using the IBM Virtual Shared Disk window and the SP Perspectives GUI, please see the online help and the “Using the SP Perspectives GUI” chapter in the *IBM Parallel System Support Programs for AIX: Administration Guide*. For information about the IBM Virtual Shared Disk, see *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

spvsd

Location

/usr/lpp/ssp/bin/spvsd

Related Information

You can access the IBM Virtual Shared Disk window by using the SP Perspectives Launch Pad. The **perspectives** command invokes the launch pad. Other Perspectives windows may be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, and **spsyspar**.

IBM Parallel System Support Programs for AIX: Managing Shared Disks.

Examples

1. To invoke the **spvsd** window, enter:

   ```
   spvsd
   ```

2. To force **spvsd** to display bold text regardless of what is set in the preference files, enter:

   ```
   spvsd -fontBold
   ```

startvsd

Purpose

startvsd – Makes an IBM Virtual Shared Disk available and activates it.

Syntax

startvsd [–**p** | –**b**] –**a** | *vsd_name* ...

Flags

–**p** Specifies that the primary server node defined for the global volume group is to be the active server.

This option is only used by IBM Recoverable Virtual Shared Disk product.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

–**b** Specifies that the secondary server node defined for the global volume group is to be the active server.

Note: This flag is used only by the IBM Recoverable Virtual Shared Disk product.

–**a** Specifies that all IBM Virtual Shared Disks that have been defined are to be started.

Operands

vsd_name Specifies an IBM Virtual Shared Disk

Description

The **startvsd** command makes the specified IBM Virtual Shared Disks available and activates them. It is equivalent to running the **preparevsd** command followed by the **resumevsd** command on the specified IBM Virtual Shared Disk.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Start an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/startvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

1. If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

   See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

2. The –**b** flag is used only by the IBM Recoverable Shared Disk product.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

Examples

To make available and activate the IBM Virtual Shared Disk **vsd1vg1n1**, enter:

```
startvsd vsd1vg1n1
```

statvsd

Purpose

statvsd – Displays IBM Virtual Shared Disk device driver statistics of a node.

Syntax

statvsd

Flags

None.

Operands

None.

Description

The **statvsd** command displays the level of IBM Virtual Shared Disk parallelism, several IBM Virtual Shared Disk IP device driver counters, nonzero sequence numbers and the nodes they are for, the node out cast status, and the *max_IP_msg_size*. If either the expected or outgoing sequence number for a node is nonzero, both are displayed along with the node number. If both the expected and outgoing sequence numbers are zero, no sequence numbers are displayed for that node.

The level of IBM Virtual Shared Disk parallelism defaults to **9** and is set via the **ctlvsd** −**p** command.

Sequence numbers are initially all zero at the first **cfgvsd**. They are incremented as requests are sent (outgoing) and received (expected), and reset via **ctlvsd** −**R** | −**r**.

The counters all start at zero at the first **cfgvsd**, then are incremented as the events they count occur. Use **suspendvsd** and **stopvsd** to ensure that there is no IBM Virtual Shared Disk activity; then use **ctlvsd** to reset the counters.

The requests queued waiting for a request block, pbuf, cache block, and buddy buffer counters indicate shortages of these resources. All four are tunable values on the **vsdnode** command. If a significant increase in these counters occurs during the running of a critical application, see *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for information about tuning IBM Virtual Shared Disk performance and how to respond to resource shortages.

When a user buffer address is not on a page boundary, two IBM Virtual Shared Disks can share a page in I/O requests. Typically, when a local IBM Virtual Shared Disk server is copying data to the user buffer, the DMA hides the page. If the client receives the data from a remote IBM Virtual Shared Disk server, network protocol interrupts the local I/O; however, the page is still hidden by the DMA. Therefore, the IBM Virtual Shared Disk places the remote request on a Rework_Q, swaps control to the local I/O, and later performs rework by copying data from the network protocol mbuf to the user buffer.

A request, or its corresponding response, may be lost due to transmission failure or failure to allocate an mbuf. The current IBM Virtual Shared Disk communication protocol implements an exponential back-off retransmission strategy. A request is retransmitted to the server a fixed number of times. The IBM Virtual Shared Disk IP device driver waits about 2 seconds for a response after initially sending the request before retransmission. Thereafter, it waits about twice as long as the last time as it cycles through the fixed number of retries. If a response is not received after the timeout expires on the last retransmission attempt, the request fails with the ETIMEOUT errno value. Currently, the sum total of retransmission time is about 15 minutes. If a request is not responded to after about 10 transmissions of the request over a 15 minute period, the request is failed. **statvsd** displays the number of requests failed due to timeouts in the timeouts counter. The retries counters display the number of requests that have been retransmitted for each retry bucket value. The number of numbers on the retries line of **statvsd** output indicates the fixed number of times a request is retransmitted. The total retries is the sum of the retries bucket values, which is the total number of request retransmissions.

There is no tuning that can be performed to affect these values; the values are provided for information only. If a request to an IBM Virtual Shared Disk is being retransmitted, and the IBM Virtual Shared Disk is suspended and subsequently resumed, the request starts over with a fresh 15 minute retry period.

If a server gets heavily loaded, it may not be able to respond to a request fast enough to prevent the client from retransmitting the request. If the server responds after the client has retransmitted the request, the client rejects the response since its sequence number no longer matches the current sequence number of the request. The client records this event in the rejected response counter that **statvsd** displays.

If a server receives a request with an unexpected sequence number, a rejected request event is recorded in a counter that **statvsd** displays.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands:

netstat −**m** for **mbuf** usage information.

/etc/no −**o thewall=16,384** to dynamically add more **mbufs**.

/etc/no −**a** | **grep thewall** to show your current **mbuf** setting.

cfgvsd, **statvsd**, **ucfgvsd**, **vsdnode**.

ctlvsd to cast nodes in and out, resetting sequence numbers and setting the *max_IP_msg_size*.

Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for information on tuning IBM Virtual Shared Disk performance and sequence numbers.

Examples

The following example displays IBM Virtual Shared Disk device driver statistics:

```
VSD driver: IP interface

          9 vsd parallelism
      24576 vsd max IP message size
          0 requests queued waiting for a request block
          0 requests queued waiting for a pbuf
          0 requests queued waiting for a cache block
          0 requests queued waiting for a buddy buffer
        0.0 average buddy buffer wait_queue size
          0 rejected requests
          0 rejected responses
          0 rejected no buddy buffer
          0 requests rework
          0 indirect I/O
          0 64-byte unaligned reads
          9 comm. buf pool shortage
          0 timeouts
retries:  0 0 0 0 0 0 0 0 0
          0 total retries
Non-zero  Sequence numbers
 node#     expected     outgoing     outcast?     Incarnation: 0

 3 Nodes Up with zero sequence numbers: 256 1000 2000
```

stopvsd

Purpose

stopvsd – Makes an IBM Virtual Shared Disk unavailable.

Syntax

stopvsd –**a** | *vsd_name* ...

Flags

–**a** Specifies that all IBM Virtual Shared Disks in the suspended state are to be stopped.

Operands

vsd_name Specifies an IBM Virtual Shared Disk. If the IBM Virtual Shared Disk is not is the suspended state, you get a failed error message.

Description

The **stopvsd** command brings the specified IBM Virtual Shared Disks from the suspended state to the stopped state. They become unavailable. All applications that have outstanding requests for the IBM Virtual Shared Disk see these requests terminate with error. Read and write requests fail with **errno** set to **ENODEV**. If the IBM Virtual Shared Disk is in the stopped state, this command leaves it in the stopped state.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Stop an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/stopvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **suspendvsd**, **ucfgvsd**

Examples

To bring the IBM Virtual Shared Disk **vsd1vg1n1** from the suspended state to the stopped state, enter:

```
stopvsd vsd1vg1n1
```

supfilesrv daemon

Purpose

supfilesrv – Is the daemon that serves the file collections on the SP system.

Syntax

startsrc –s supfilesrv

stopsrc –s supfilesrv

Flags

There are no flags or options for this daemon. All arguments and options are defined to the System Resource Controller (SRC).

Description

This daemon executes on the control workstation and the boot and install servers. It responds to requests executed on the nodes to update collections of files configured for File Collections. The **supfilesrv** daemon is under control of the System Resource Controller (SRC). It is normally started as part of the SP initialization scripts. Under normal circumstances, the administrator does have to start or stop the daemon.

Exit Values

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on file collections.

Location

/var/sysman/etc

Related Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on file collections, the System Resource Controller, and user management.

Examples

1. To start the **supfilesrv** daemon, enter:

   ```
   startsrc -s supfilesrv
   ```

2. To stop the **supfilesrv** daemon, enter:

   ```
   stopsrc -s supfilesrv
   ```

supper

Purpose

Manages the SP file collections.

Syntax

supper [–**d**] [–**o** *options*] [–**v**] *subcommands*

Flags

–**d**	Turns on debug mode.
–**o** *options*	Lists SUP options to pass to SUP. See the SUP manual pages for more detail.
–**v**	Turns on verbose mode and echoes print of SUP messages.

Operands

Subcommands

activate *volume*	Set the active volume group. The active volume group must be set before installing a collection that requires a file system.
debug	Choice of **on** | **off**. Turn debug messages on or off.
diskinfo	Show available disk space and active volume.
files *collection*	Show all files associated with a resident collection.
install *collection*	Install a collection.
log	Show summary of last/current supper session.
offline *collection*	Disable updates of a collection.
online *collection*	Enable updates of a collection (this is the default).
quit	Exit the program.
remove *collection*	Remove a collection.
reset *collection*	Set the last update time of a collection to the epoch.
rlog	Show raw output of last/current supper session.
scan *collection*	Run a scan for a collection.
serve	List all collections this machine is able to serve.
status	Show the current status of all available collections. The status information includes the names of all collections, whether they are resident on the local machine, and the name and size of the file system associated with each collection.
update *collection*	Update a collection.
verbose	Choice of **on** | **off**. Turn SUP output messages on or off.
when	Print the last update time of all resident collections.
where	Show current servers for collections.

! *command* Shell escape.

Description

You can invoke **supper** as an interactive session by entering the **supper** command without any parameters or subcommands. This allows you to enter the subcommands in an interactive dialog.

Examples

```
/var/sysman/supper rlog
/var/sysman/supper status
/var/sysman/supper scan user.admin
/var/sysman/supper update power_system
```

suspendvsd

Purpose

suspendvsd – Deactivates an available IBM Virtual Shared Disk.

Syntax

suspendvsd –**a** | *vsd_name*...

Flags

–**a** Specifies that all the IBM Virtual Shared Disks in the active state are to be suspended.

Operands

vsd_name Specifies an IBM Virtual Shared Disk. If the IBM Virtual Shared Disk is not in the active state, you get a failed error message.

Description

The **suspendvsd** command brings the specified IBM Virtual Shared Disks from the active state to the suspended state. They remain available. Read and write requests which were active while the IBM Virtual Shared Disk was active are suspended and held. Subsequent read and write operations are also held. If the IBM Virtual Shared Disk is in the suspended state, this command leaves it in the suspended state.

If you issue this command for the server node and not the client, retries occur with eventual failure. Failure occurs within 15 minutes.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Suspend an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/suspendvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **ucfgvsd**

Examples

To bring the IBM Virtual Shared Disk **vsd1vg1n1** from the active state to the suspended state, enter:

```
suspendvsd vsd1vg1n1
```

sysctl

Purpose

sysctl – Is the command interface to Sysctl remote command execution and monitoring server.

Syntax

sysctl [–**c** *collection_of_nodes*] [–**f** *num*] [–**h** *host*] [–**L**] [–**m**] [–**n**] [–**P** *port*] [–**q**] [–**r** *file* | –] [–**s**] [–**t** *sec*] [–**T** *sec*] [–**v**] [–**x**] [*command ...*]

Flags

[–c *collection_of_nodes***]**
Runs the command on the specified *collection_of_nodes*. If the *collection_of_nodes* argument contains a "/", it is assumed to name a file that holds the names of host machines. More than one *collection_of_nodes* can be specified with multiple –**c** options. Use – for standard input.

[–f *num***]** Controls maximum fan-out, when multiple hosts are specified. By default, a maximum of eight concurrent connections are used. The most allowed is 128. You may want to increase *num* to allow for greater parallelism when running commands that place low demand on system services.

[–h *host***]** The server on which to execute the command. More than one host can be specified using multiple –**h** options. If no –**h** (or –**c**) option is specified, the server is assumed to be the local host.

[–L] Provides an alternate way to delimit output from multiple servers. A host name, such as **helloworld.testcell.ibm.com** precedes each line of output from the host. This type of output is easier to parse in programs and scripts than the default. The default format is to separate the output in blocks labeled with the host name. This is compatible with the **dshbak** output filter.

[–m] Uses safe (message-authenticated) communication.

[–n] Sends no authentication information.

[–P *port***]** Connects to the server using a specified port number. If this flag is not specified, the port is obtained from file **/etc/services**.

[–q] Quick mode—do not wait for the result from the server. In this case, the server does not return the result to the client.

[–r *file* | –**]** Replays (drops) this file on the servers. Used to pass scripts to **sysctl** for execution. You can specify standard input as the source of input by entering a dash (–) instead of a file name; press <**Ctrl-D**> when you are finished entering script commands from standard input.

[–s] Tells the server to send results back via a TCP socket. The output from the server is demultiplexed into standard output and standard error, allowing you to separate the two types of output.

Using a socket also enables two-way communication between the client and a command procedure. Any commands that start a dialog requiring input from the client must be run with the **–s** option.

[–t *sec***]** Specifies the connection timeout in seconds. The default is 10 seconds.

[–T *sec***]** Specifies the maximum time to wait in seconds for the result from a **sysctl** command. The default is 30 minutes.

[–v] Prints the version number of **sysctl**, then exits.

[–x] Sends a NULL RPC (such as a ping) to the servers. You can use this to check if a server is up before you try to run a command on that machine.

Operands

command ...
The command to pass to the server. You can enter multiple commands.

If you do not specify a command, **sysctl** runs in interactive mode.

Description

The **sysctl** program provides a simple command line interface for communicating with the **Sysctl** server, **sysctld**. Together, the **Sysctl** client and server provide monitoring and execution abilities needed to remotely manage workstations on all nodes on the SP system. **Sysctl** connects to a remote host's **sysctld** using TCP/IP, passes keywords and commands to the server, and writes any output returned to standard output. **sysctl** does not interpret the commands it passes.

The **Sysctl** server uses the Tcl embeddable command language as the foundation for its built-in interpreter. The server is augmented with application-specific commands that may vary between servers. The **Sysctl** (Tcl) expressions that are passed to the server for execution can be anything from a single command to an entire script. The server uses SP authentication services for reliable third party authentication. The **Sysctl** request sent from the client optionally contains an authentication ticket that uniquely identifies the user that initiated the request. The server's built-in authorization mechanism controls the set of commands available to a user.

Files

/etc/krb-srvtab Contains authentication key for services.

/etc/sysctl.acl Default ACL file for the **sysctl** server

/etc/sysctl.conf
sysctl server configuration file.

Related Information

Commands: **dshbak**, **hostlist**, **sysctld**

The chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Examples

These examples show commands issued from the shell prompt. Unless noted otherwise, the syntax is the same (minus the word **sysctl**) when issuing the command from within an interactive session of **sysctl**:

1. To list the local file systems on server **ceti-alpha5**, enter:

   ```
   sysctl listfs -h ceti-alpha5
   ```

2. To list all the commands that you are authorized to run on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 info commands
   ```

3. To add principal **arielle.admin** to the ACL of trusted users for the **sysctl** server on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 acladd -p arielle.admin
   ```

   If you do not specify a realm, the local realm is assumed. If **ceti-alpha5** is in realm **TESTCELL.HAL.COM**, **sysctl** returns the following message:

   ```
   -principal arielle.admin@TESTCELL.HAL.COM
   ```

4. To add principal **arielle.admin** to ACL file /**test**/**data**/**mount.acl** on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 acladd -f /test/data/mount.acl \
   -p arielle.admin
   ```

5. To see if **arielle** is in ACL file /**test**/**data**/**mount.acl** on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 aclcheck -f /test/data/mount.acl arielle
   ```

   The server returns **1** if **arielle** is in the ACL, **0** if it is not.

6. To check to see if you are authorized to run the **test:mount_if** command on the SP node **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 checkauth -cmd test:mount_if
   ```

   The server returns **1** if you are authorized, **0** if not.

sysctld daemon

Purpose

sysctld – Contains the remote execution and monitoring server.

Syntax

sysctld [–**sd**] [–**a** *acl*] [–**f** *file*] [–**k** *keyfile*] [–**l** *log_file*] [–**P** *port*]

Flags

–**d**	Runs in debug mode. This causes a large amount of debugging information to be written to the log file as the server executes requests.
–**s**	Runs in security audit mode. An audit trace is written to the log file as the server executes each user request.
–**a** *acl*	Specifies the Access Control List (ACL) file. The default ACL file is **/etc/sysctl.acl**.
–**f** *file*	Specifies an alternate configuration file. The default is **/etc/sysctl.conf**.
–**k** *keyfile*	Specifies the server's key file. The default key file is **/etc/srvtab**.
–**l** *log_file*	Specifies the server log file. The default log file is **/var/adm/SPlogs/sysctl/sysctld.log**. (This is lowercase **l**, as in **l**ist.)
–**P** *port*	Specifies the server port number. If this flag is not specified, the port is obtained from **/etc/services** using **sysctl/tcp** as the key.

Operands

None.

Description

The **sysctld** daemon is the server component for the Sysctl remote command execution facility. Security and performance characteristics of **sysctld** make it an ideal mechanism for managing a large, distributed computing environment. Typically, one instance of **sysctld** runs on every workstation. Commands are sent to a **sysctld** server via the **sysctl** client program. When a command request is received, **sysctld** parses and executes the request using an embedded Tcl command interpreter. Authorization to execute commands is determined by *authorization callbacks* that are attached to each command. These callbacks are pieces of Tcl code that implement the security policy for the **sysctld** server.

Security and Access Control

Sysctl uses the SP authentication service for reliable third-party authentication. Command requests sent by the client include an authentication ticket giving the identity of the client. As the server executes the commands sent by the client, the client's access to commands and variables is determined dynamically via *authorization callbacks*. In a typical command language, a *procedure* has a name, a set of arguments, a set of commands which form the body of the procedure, and a return value. With Sysctl procedures, an additional attribute is added, a policy for determining who is able to run the command. These policies are implemented

using authorization callbacks. An authorization callback is a piece of Tcl code that is *attached* to a command and determines the access policy for that command. The authorization callback for a command is invoked whenever a client attempts to execute the command. If the callback returns a Tcl error, the command is not executed and the following message is returned to the caller:

```
Authorization Denied
```

If the callback returns a normal Tcl result, the command executes normally.

Sysctl variables also have an authorization callback attached to them which determines the read-access policy for the variable. Therefore, it is possible to create a "private" variable in which you restrict the set of clients who have access to its value.

The **sysctld** server defines a set of commands which are designed to be used as authorization callbacks. These commands provide a simple authorization policy; if more complex authorizations are required, you have the ability to code your own authorization callback procedures.

NONE Always returns a normal result. Any command it is attached to can be executed by any user.

AUTH Returns a normal result if the user is authenticated via SP authentication services.

ACL [*file*] Returns a normal result if the user is authenticated as the local host principal or if the user is listed in the ACL passed as an argument. If an ACL file is not supplied, the default system ACL (defined by the $ACL variable) is used. See **sysctl.acl** for more details about Sysctl ACLs.

SYSTEM Always returns a Tcl error. Any command it is attached to can never be executed directly by a user. Instead, these commands are intended to be executed from within a procedure registered with the Sysctl server by way of its configuration file.

Bypassing Authorization Callbacks

Under certain circumstances, the authorization callbacks are bypassed by the server. While the checks are bypassed, the user has access to all commands (and r/w access to variables) defined in the server. The authorization callbacks are bypassed while the server is:

- Reading configuration files
- Executing an authorization callback
- Executing the body of a procedure for which a user is authorized to run

Since authorization checks are bypassed during procedure execution, users are able to execute procedures that contain commands they are not authorized to run directly. For example, if a variable has the SYSTEM callback attached to it, a procedure for which a user is authorized can reference or modify the variable even though that user cannot modify or reference it.

Authorization Variables

Several variables, accessible as read-only variables in the interpreters or as environment variables, are set by the server prior to executing the client request. These variables provide a mechanism for external commands and procedures to perform their own authorization checking independent of the server's standard authorization checks. They are:

SCHOST Specifies the name of the host from which the request was issued.

SCPRINCIPAL
Indicates the authenticated identity of the issuer. If the user is unauthenticated, it is set to NULL.

SCUSER Specifies the base name of the user, or NULL if unauthenticated.

SCINSTANCE
Specifies the instance of the user, or NULL if unauthenticated.

SCREALM Specifies the realm of the user, or NULL if unauthenticated.

SCLHOST Specifies the host name of the local server.

SCLREALM Specifies the local realm of the server.

SCLPRINCIPAL
Specifies the principal name of the local **sysctld** server.

SCMODE Specifies the communication mode between the client and server, either WAIT, NOWAIT, or SOCKET. NOWAIT indicates that the client is not interested in the result of the operation, and it is discarded by the server. SOCKET indicates that the result of the operation is returned to the client via a TCP/IP socket.

Determining Connection Authorization Policy

Whenever a client connects to the server, the command **svcconnect** is invoked. If the user is not authorized to run this command or if the command returns a Tcl error, the result of the command is returned to the user and the connection is broken. Therefore, the **svcconnect** command determines the connection policy for the server. By default, **svcconnect** simply returns a normal result and its authorization callback is AUTH. This implies that any authenticated user can connect. This policy can be altered by changing the authorization for the **svcconnect** callback (via the **setauth** command), or by redefining the **svcconnect** procedure (via the **create proc** command).

Server Configuration

At start up, the server reads a configuration file. By default, this file is named **/etc/sysctl.conf**. The **–f** flag is used to specify an alternate configuration file. The server interprets the contents of the configuration file as Tcl commands. Typically, additional commands and variables are defined in this file. Also, commands are available that instruct the server to read additional configuration files or dynamically load in shared libraries. In this way, the set of commands available to a **sysctld** server is extendable. Refer to the **sysctl.conf** file for more details.

Signals

Sending a SIGHUP signal to the server causes it to close the log file, delete and re-create all interpreters, reread the configuration files, and reinitialize the log file. The **svcrestart** command performs the same function.

Logging

The **sysctld** default log file is **/var/adm/SPlogs/sysctl/sysctld.log**. This default can be overridden with the **–l** command line option, or by by setting the LOG variable in the Sysctl configuration file. Each time a request is received, the **svclogevent** Sysctl command is invoked. By default, it writes a record to the log file giving the identity of the user who sent the request or "unknown" if the user is not authenticated. A different logging policy can be achieved by redefining the **svclogevent** procedure in the server's configuration file.

While running in security audit mode, each line written to the log file is tagged with a *Connection ID* field which is used to filter the audit trail for a particular connection in cases where multiple connections are processed simultaneously.

Starting and Stopping the sysctld Daemon

The **sysctld** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **sysctld** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **sysctld**. In order to start the **sysctld** daemon, use the **startsrc –s sysctld** command. This starts the daemon with the default arguments and SRC options. The **sysctld** daemon is setup to be respawnable and be the only instance of the **sysctld** daemon running on a particular node or control workstation. Do **not** start the **sysctld** daemon from the command line without using the **startsrc** command to start it.

To stop the **sysctld** daemon, use the **stopsrc –s sysctld** command. This stops the daemon and does not allow it to respawn.

To display the status of the **sysctld** daemon, use the **lssrc –s sysctld** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4 Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget –q "subsysname=sysctld" SRCsubsys** command.

Files

/etc/sysctl.acl The default ACL used to assign base authorizations.

/etc/sysctl.conf
The default configuration file read by the server on start up.

/var/adm/SPlogs/sysctl/sysctld.log
The default log file.

Related Information

Command: **sysctl**

Files: **sysctl.acl**, **sysctl.conf**

Examples

1. To start the **sysctld** daemon, enter:

   ```
   startsrc -s sysctld
   ```

2. To stop the **sysctld** daemon, enter:

   ```
   stopsrc -s sysctld
   ```

3. To display the status of the **sysctld** daemon, enter:

   ```
   lssrc -s sysctld
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **sysctld** daemon, enter:

   ```
   odmget -q "subsysname=sysctld" SRCsubsys
   ```

SYSMAN_test

Purpose

SYSMAN_test – Verifies that the installation and customization of the Systems Management components of the SP system completed successfully.

Syntax

SYSMAN_test [–**q** | –**v**] [–**l** *log_file*]

Flags

–**q** Specifies quiet mode; suppresses all but summary output to standard output.

–**v** Specifies verbose mode, includes informational messages to standard output.

–**l** *log_file* Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.)

Operands

None.

Description

The **SYSMAN_test** command performs various tests to determine whether the systems management components of the SP system are completely installed and customized properly.

A return code of 0 indicates that the test completed as expected; otherwise it returns the number of failures. If you do not specify the –**q** flag, a message is displayed on standard output that indicates the success or failure of the tests. In either case, the command returns 0 if successful, 1 if not. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/SYSMAN_test.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the RS/6000 SP System Management option.

Files

/usr/lpp/ssp/bin/SYSMAN_test
Path name of this command.

/var/adm/SPlogs/SYSMAN_test.log
Default log file.

Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **spmon_ctest**, **spmon_itest**

Examples

To verify systems management following customization, saving error messages in **sm.errors** in the current working directory, enter:

```
SYSMAN_test -l sm.errors
```

syspar_ctrl

Purpose

syspar_ctrl – Starts, stops, adds, deletes, and refreshes the system partition-sensitive subsystems installed on your SP system.

Syntax

syspar_ctrl [–**G**] [–**V**] {–**a** | –**d** | –**s** | –**k** | –**t** | –**o** | –**c** | –**r** | –**h** | –**A** | –**D** | –**E** | –**R**} [*subsystem_name*]

Flags

–**h** (help) Displays usage information. If a *subsystem_name* is specified, help is provided only for the specified subsystem's control script. Help is displayed as a syntax description and is written to standard output. Once help is displayed, no other action is taken even if other valid options are entered with the –**h** flag.

–**a** (add) Adds all subsystems. If a *subsystem_name* is specified, only the specified subsystem is added. The –**a** flag invokes each subsystem's control script. Typically, this causes each subsystem's control script to add itself to the System Resource Controller (SRC) subsystem, /**etc**/**inittab** and /**etc**/**services**. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

–**A** (add and start) Adds and starts all subsystems. If a *subsystem_name* is specified, only the specified subsystem is added and started. Each subsystem's control script is invoked with the –**a** flag followed by the –**s** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the –**a** flag followed by the –**s** flag.

–**c** (clean) Cleans up after all of the subsystems. If a *subsystem_name* is specified, only the specified subsystem is cleaned up. Each subsystem's control script is invoked with the –**c** flag. Typically, this causes each subsystem's control script to stop any subsystem daemons that may be running and clean or remove all entries for this subsystem from the SRC, /**etc**/**inittab**, /**etc**/**services**. This flag is similar to the –**d** (delete) flag, but independent of system partitions. Cleaning up the subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller subsystems file. You can use this option to clean up subsystem information while trying to get back to some preexisting state, such as when an old System Data Repository (SDR) is restored and the old system partitioning needs to be restored.

–**d** (delete) Deletes all subsystems. If a *subsystem_name* is specified, the specified subsystem is deleted. Each subsystem's control script is invoked with the –**d** flag. Typically, this causes each subsystem's control script to delete itself from the SRC subsystem, /**etc**/**inittab** and /**etc**/**services**. Deleting subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller subsystems file. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

–D (stop and delete) Stops and deletes all subsystems. If a *subsystem_name* is specified, that subsystem is stopped and deleted. Each subsystem's control script is invoked with the **–k** flag followed by the **–d** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the **–k** flag followed by the **–d** flag.

–E (examine) Examines all subsystems. If a *subsystem_name* is specified, the specified subsystem is examined in the Syspar Controller subsystems file. Each subsystem name - control script pair in the subsystems file is examined and displayed. Entries that are not valid are noted. An entry is not valid when the control script for a particular subsystem does not exist at the specified location or does not have the correct read and execute permissions.

–G (global) Invokes the appropriate underlying subsystem's control scripts for each system partition. If the **–G** flag is not specified, the appropriate underlying subsystem's control script is run only in the current system partition (SP_NAME).

–k (kill or stop) Stops all subsystems. If a *subsystem_name* is specified, only the specified subsystem is stopped. Each subsystem's control script is invoked with the **–k** flag. Typically, this causes each subsystem's control script to stop any daemons associated with this particular subsystem. Stopping subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller's subsystem file. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

–r (refresh) Refreshes all subsystems. If a *subsystem_name* is provided, only the specified subsystem is refreshed. Each subsystem's control script is invoked with the **–r** flag. Typically, this causes each subsystem's control script to rebuild configuration data and refresh any daemons associated with this particular subsystem. Subsystems may need to be refreshed when nodes are added to an existing system or the nodes PSSP version changes from PSSP 2.1 or 1.2 to PSSP 2.2 or vice versa. The actual function that is performed depends on the subsystem. This option is only meaningful when run on the control workstation.

–R (restore) Restores all subsystems. If a *subsystem_name* is specified, only the specified subsystem is restored. All subsystems are stopped and deleted before they are added and started. Each subsystem's control script is invoked with the **–k** flag followed by the **–d** flag, then the **–a** flag followed by the **–s** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the **–D** flag followed by the **–A** flag.

–s (start) Starts all subsystems. If a *subsystem_name* is specified, only the specified subsystem is started. Each subsystem's control script is invoked with the **–s** flag. Typically, this causes each subsystem's control script to start any daemons associated with this particular subsystem. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

–t (trace on) Turns the trace option on for all subsystems. If a *subsystem_name* is specified, the trace option is turned on only for the specified subsystem. Each subsystem's control script is invoked with the **–t** flag.

Note: We recommend only turning on a particular subsystem's trace by providing a subsystem name. If the trace is turned on for all subsystems, the volume of data produced may quickly fill up /**var**.

–o (trace off) Turns the trace option off for all subsystems. If a *subsystem_name* is specified, the trace option is turned off only for the specified subsystem. Each subsystem's control script is invoked with the **–o** flag.

–V (verbose) Turns verbose mode on in the **syspar_ctrl** script which then prints out the actual calls it makes to the underlying subsystem control scripts. It also prints out additional information that is useful for debugging.

Operands

subsystem_name

Specifies the subsystem name that you want the command to act on. If a *subsystem_name* is not provided, this command is run for all subsystems that are listed in the Syspar Controller subsystems file (syspar_subsystems). For example, if you only want this command to work with the Event Management subsystem, enter:

```
syspar_ctrl option haem
```

Description

This command acts as an interface to the system partition-sensitive subsystems supporting the functions that are shared by all subsystems. This command is also referred to as the Syspar Controller. It can be used to add or delete, start or stop, refresh or restore the subsystems, and various other functions. When used on the control workstation, it works with the subsystems on the control workstation. When used on the nodes, it works with the subsystems on the nodes. The refresh option is an exception. In order to refresh some subsystems, the subsystem must be refreshed on both the control workstation and on the nodes. In this case, the refresh on the control workstation will **dsh** an appropriate refresh command from the control workstation to the appropriate nodes.

This command supports two types of options: primitive options and macro options. Primitive options are passed directly to the underlying control scripts, for example, **–a** (add), **–d** (delete), **–r** (refresh). Macro options conveniently group a commonly used set of primitive options into one option, for example, **–R** (restore). All of the subsystems and each subsystem's control script that are managed by the Syspar Controller are listed in the Syspar Controller subsystems file. By default, all of the control scripts listed in the Syspar Controller subsystems file will be called unless a *subsystem_name* is provided. In that case, the control script for just the specified subsystem will be called.

This command is automatically called when the system is partitioned (**spapply_config**) to first stop and delete the system partition-sensitive subsystems from system partitions that are being removed, and then to add and start the

system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) in new system partitions.

The Syspar Controller is also called when restoring the SDR with **sprestore_config** to first clean up and then add and start the system partition-sensitive subsystems (for example, **hats**, **hb** and **hr**) in each system partition.

The Syspar Controller also needs to be called with refresh flag (–**r**) by the System Administrator using the command line whenever a node is added or deleted from the system, or a node is migrated between PSSP 1.2 or 2.1 to PSSP 2.2.

Files

syspar_subsystems
: Lists all of the system partition sensitive subsystems and their control scripts that are controlled by the Syspar Controller. Only the **syspar_ctrl** command should read this file. This file is located in the directory **/usr/lpp/ssp/config/cmi**.

Security

You must be running with an effective user ID of root.

Environment Variables

SP_NAME
: **syspar_ctrl** sets the SP_NAME environment variable prior to calling the underlying subsystems. Typically, SP_NAME is set to the value returned from the **spget_syspar -n** command. However, when **syspar_ctrl** is called with the –**G** flag, **syspar_ctrl** sets SP_NAME in turn to each value returned by the **splst_syspars -n** command. The –**c** flag ignores system partition boundaries while all other options respect system partition boundaries.

Exit Values

0
: Indicates the successful completion of the command.

1
: Indicates that the command failed. Most likely a subsystem's control script returned a bad return code.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/syspar_ctrl

Related Information

Commands: **emonctrl**, **hatsctrl**, **hbctrl**, **hrctrl**, **haemctrl**, **hagsctrl**, **pmanctrl**, **sp_configdctrl**, **spapply_config**, **spcw_apps**, **sprestore_config**

Examples

1. To add and start all of the system partitions subsystems in each of the system partitions, enter:

   ```
   syspar_ctrl -G -A
   ```

2. To stop and delete all of the system partition subsystems in each of the system partitions, enter:

   ```
   syspar_ctrl -G -D
   ```

3. To refresh all of the system partition subsystems in the current system partition, enter:

   ```
   syspar_ctrl -r
   ```

4. To restore all of the system partition subsystems running in the current system partition, enter:

   ```
   syspar_ctrl -R
   ```

5. To stop all of the system partition subsystems running in the current system partition, enter:

   ```
   syspar_ctrl -k
   ```

6. To get help for the event manager subsystem (haem) control script, enter:

   ```
   syspar_ctrl -h haem
   ```

7. To display a list of all subsystems managed by the Syspar Controller, enter:

   ```
   syspar_ctrl -E
   ```

8. To see the state of the system partition subsystems controlled by the Syspar Controller for system partition spp1, enter the commands:

   ```
   lssrc -a | grep spp1
   lssrc -a | grep sp_configd
   ```

   Note: The SDR is not managed by the System Controller.

sysparaid

Purpose

sysparaid – Creates a layout for a new system partition configuration of an SP system.

Syntax

sysparaid [–**h**] [–**i**] [–**s** *layout_name* | *a_fully_qualified_path*]
[–**t** *tmpdir*] *input_file* [*topology_file*]

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description of the command and the startup guidelines are displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**i** Creates a switch-map file (**spa.sysinfo**) for the current SP system if a System Data Repository (SDR) is present. If an SDR is not available, it will generate the file for the system described by the *topology_file*. If the –**s** flag is entered along with the –**i** flag, it will be ignored.

–**s** Saves the newly generated layout in the system partition directory tree (/**spdata**/**sys1**/**syspar_configs**/**...**) if a *layout_name* is specified; otherwise, it will be saved under the directory specified by the fully qualified path given.

–**t** Saves the snapshot, performance, and intermediate files under *tmpdir*.

Operands

input_file Specifies the file containing the system partition configuration requirement.

topology_file
Specifies the topology file for the system to be partitioned. This operand is required only when the topology file for the system to be partitioned is not under /**spdata**/**sys1**/**syspar_configs**/**topologies**. It is also required with the –**i** flag when there is no SDR or when the switch-map file for a system not represented by the SDR is desired.

Description

Use this command to invoke the System Partitioning Aid, a tool for generating new system partition configuration layouts. When invoked with the –**i** flag, it creates a switch-map file that will help the user to generate the *input_file* for creating a layout for a desired system partition configuration. When invoked with no flags or with the –**s** or –**t** flags, it attempts to partition the system according to the input requirement. If the attempt is unsuccessful, it will output appropriate error messages to the log and exit. If the attempt is successful and the –**s** flag is specified, the newly created layout will be saved at the desired location specified by the flag argument.

Standard Input

This command requires an input file when invoked with no flag or the –**s** flag. The template for the input file can be found in **/spdata/sys1/syspar_configs/bin**.

Standard Output

Informational messages are written to standard output.

Standard Error

Error messages are written to standard error.

Output Files

This command creates **spa.snapshot** and **spa.metrics** under *tmpdir* (if specified) or under the current working directory. If the –**s** flag is specified and the attempt is successful, it creates the following under the layout directory:

layout.desc

spa.snapshot

nodes.syspar and a system partition directory for each system partition in the layout. Under each system partition directory, it creates the following:

node list
topology
spa.snapshot
spa.metrics

When invoked with the –**i** flag, the command creates **spa.sysinfo** under *tmpdir* (if specified), or under the current working directory.

Extended Description

The **sysparaid** command uses a set of built-in rules to create a layout for a desired system partition configuration. The following startup guidelines will help to generate an acceptable input to the command:

1. The nodes can be identified by either using *node_numbers* or *switch_port_numbers*. While both schemes are permitted for partitioning a system defined by an SDR, *switch_port_numbers* is the only allowed choice when running the tool without an SDR. Also, the numbering schemes cannot be mixed when both schemes are allowed.
2. Identify the four nodes linked to any switch chip to place them in the same system partition. If an SDR is present, the identity of the switch chips linked to the nodes in the system can be obtained by issuing the following command:

   ```
   sysparaid -i -t spa_dir
   ```

   This command places a **spa.sysinfo** file in the **spa_dir** if the –**t** flag is used; otherwise, it places it in the current directory. If an SDR is not present, issue the following command:

   ```
   sysparaid -i -t spa_dir topology_file
   ```

   where *topology_file* is the name of the topology file for the system to be partitioned.

   Note: In this case, only the *switch_port_numbers* are provided. No *node_numbers* are available.

3. The keyword “remaining_nodes” can be used for the last system partition provided all nodes or switch ports not in the last system partition were placed in other system partitions. Therefore, the keyword cannot be used with the *node_number* numbering scheme for systems with empty input switch ports.
4. Nodes on a switch board can be part of a maximum of two multichip system partitions.
5. The input file must be formatted according the the template provided in **/spdata/sys1/syspar_configs/bin/inpfile_template**.

Security

Any user can run this command. Only users authorized to write to the system partitioning directory can save a generated layout under it.

Location

/usr/lpp/ssp/bin

Related Information

The **spsyspar** command provides the graphical user interface (GUI) for the System Partitioning Aid.

Examples

1. The following is an example of an input file with the switch port number option (all switch ports linked to nodes):

   ```
   Number of Nodes in System: 32
   Number of Frames in System: 2
   Frame Type: tall
   Switch Type: HiPS
   Number of Switches in Node Frames: 2
   Number of Switches in Switch Only Frames: 0
   Node Numbering Scheme: switch_port_number
   Number of Partitions: 3
   Partition Name: part1
   Number of Nodes in Partition: 8
   0 - 7
   Partition Name: part2
   Number of Nodes in Partition: 8
   8 - 15
   Partition Name: part3
   Number of Nodes in Partition: 16
   remaining_nodes
   ```

   To use **/tmp** as the working directory, enter:

   ```
   sysparaid -t /tmp inpfile
   ```

   You should receive a message similar to the following:

   ```
   A layout, for the desired system partition configuration or an
   equivalent, can be created.
   To save this layout, invoke the command again with -s option.
   ```

   To save the layout for this configuration under **/spdata/sys1/syspar_configs/2nsb0isb/config.8_8_16/layout.myconfig**, enter:

```
sysparaid -s myconfig inpfile
```

To save the layout for this configuration under **/tmp/custom/config1**, enter:

```
sysparaid -s /tmp/custom/config1 inpfile
```

2. The following is an example of an input file with the switch port number option (not all switch ports in the system are linked to nodes):

```
Number of Nodes in System: 87
Number of Frames in System: 6
Frame Type: tall
Switch Type: SP
Number of Switches in Node Frames: 6
Number of Switches in Switch Only Frames: 4
Node Numbering Scheme: switch_port_number
Number of Partitions: 2
Partition Name: ProductionPartition
Number of Nodes in Partition: 82
0
4
16 - 95
Partition Name: TestPartition
Number of Nodes in Partition: 5
2
6
8
10
12
```

If you enter the **sysparaid -s myconfig inpfile** command, this configuration will be saved under **/spdata/sys1/syspar_configs/6nsb4isb/config.12_84/layout.myconfig**. Note that the nine unspecified switch port numbers have been allocated to one of the two system partitions.

3. The following is an example of an input file with the node number option (not all switch ports are linked to nodes):

```
Number of Nodes in System: 8
Number of Frames in System: 2
Frame Type: tall
Switch Type: SP
Number of Switches in Node Frames: 1
Number of Switches in Switch Only Frames: 0
Node Numbering Scheme: node_number
Number of Partitions: 3
Partition Name: part1
Number of Nodes in Partition: 2
25
29
Partition Name: part2
Number of Nodes in Partition: 4
1
5
17
21
Partition Name: part3
Number of Nodes in Partition: 2
3
7
```

This input file for a particular SP system returned the location of an existing layout:

```
The layout for the desired/equivalent system partition configuration is
under /spdata/sys1/syspar_configs/1nsb0isb/config.4_4_8/layout.2
```

4. The **spa.sysinfo** file for the system in Example 3 that was generated using the **sysparaid -i** command follows:

```
switch_number   switch_chip   switch_port_number   node_number
      1             4                 9              25
      1             4                13              29
      1             5                 0               1
      1             5                 1              17
      1             5                 4               5
      1             5                 5              21
      1             6                 2               3
      1             6                 6               7
```

5. The following is an example of an input file for a switchless system:

```
Number of Nodes in System: 32
Number of Frames in System: 2
Frame Type: tall
Switch Type: NA
Number of Switches in Node Frames: 0
Number of Switches in Switch Only Frames: 0
Node Numbering Scheme: switch_port_number
Number of Partitions: 2
Partition Name: part1
Number of Nodes in Partition: 14
2 - 5
10
11
13
15
19
24 - 25
29 - 31
Partition Name: partition2
Number of Nodes in Partition: 18
remaining_nodes
```

To save the layout for this configuration under **/spdata/sys1/syspar_configs/2nsb0isb/config.14_18/layout.myconfig**, enter:

```
sysparaid -s myconfig inpfile
```

s1term

Purpose

s1term – Opens a connection to an SP node's S1 serial port.

Syntax

s1term [–**G**] [–**w**] *frame_ID slot_ID*

Flags

–**G**	Allows specification of nodes outside the current system partition.
–**w**	Opens the connection in read/write mode.

Operands

frame_ID	Specifies the number of the frame containing the node.
slot_ID	Specifies the number of the slot containing the node.

Description

Use this command to open a connection to the S1 serial port of the SP node contained in the slot specified by the *frame_ID* and *slot_ID* operands. The specified node must be in the current system partition unless the –**G** flag is also specified. By default, the connection is read only. As data arrives from the serial port, it is written to standard output. When the connection is read/write and standard input is a terminal, the terminal is placed in *raw* mode, that is, canonical processing is turned off in the terminal driver. As data is read from standard input, it is sent to the S1 serial port. Standard input and output can be files or pipes.

When the connection is read only, the command terminates upon receipt of a signal, usually generated by the terminal Interrupt key. When in read/write mode, the command terminates when either the termination character or End-of-File is read from standard input. The termination character is **Ctrl-x** by default. Another termination character can be used by setting the S1TERMESC environment variable to the octal (denoted by leading 0), decimal or hexadecimal (denoted by leading 0x) value of the desired termination character.

Note: The termination character must only be one byte.

To execute this command, the user must be authorized to access the Hardware Monitor subsystem and, for the frame specified to the command, must be granted S1 permission. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **kinit** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **kinit**.

Files

/usr/lpp/ssp/bin/s1term
Contains the **s1term** command.

Related Information

Commands: **hmcmds**, **hmmon**

Examples

1. To open an interactive connection to the S1 serial port of the node in slot 8 in frame 12, enter:

   ```
   s1term -w 12 8
   ```

2. To write the output of the S1 serial port of the node in slot 2 in frame 9 to a file, enter:

   ```
   s1term 9 2 > s1term.output
   ```

ucfghsd

Purpose

ucfghsd – Makes a data striping device (HSD) for the IBM Virtual Shared Disks unavailable.

Syntax

ucfghsd –**a** | *hsd_name*...

Flags

–**a** Specifies that all the data striping devices defined are to be unconfigured.

Operands

hsd_name Specifies the name of a specific HSD that is unconfigured.

Description

This command unconfigures the already defined data striping devices . This command does not change the definition of the HSDs; it makes the HSDs unavailable on one node.

Files

/usr/lpp/csd/bin/ucfghsd Specifies the command file.

Security

You must have root privilege to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsd**, **defhsd**, **hsdatalst**, **lshsd**, **undefvsd**

Examples

To unconfigure the data striping device *hsd1*, enter:

```
ucfghsd hsd1
```

ucfghsdvsd

Purpose

ucfghsdvsd – Makes a data striping device (HSD) and the IBM Virtual Shared Disks that comprise it unavailable on one node.

Syntax

ucfghsdvsd –**a** | {*hsd_name*...}

Flags

–**a** Specifies that all the data striping devices defined on this system or system partition are to be unconfigured.

hsd_name Specifies the names of defined HSDs that are to be unconfigured. This command unconfigures the underlying IBM Virtual Shared Disks as well.

Operands

None.

Description

Use this command to unconfigure HSDs and their underlying IBM Virtual Shared Disks. This command does not change the definition of the HSDs and IBM Virtual Shared Disks; it just makes them unavailable.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit hsd_mgmt
```

and select the Unconfigure an HSD and its Underlying IBM Virtual Shared Disks option.

Files

/usr/lpp/csd/bin/ucfghsdvsd Specifies the command file.

Security

You must have root privilege or **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfghsdvsd**, **ucfghsd**, **ucfgvsd**

Examples

To unconfigure the data striping device *hsd1* and the IBM Virtual Shared Disks that comprise it, enter:

```
ucfghsdvsd hsd1
```

ucfgvsd

Purpose

ucfgvsd – Makes an IBM Virtual Shared Disk unavailable.

Syntax

ucfgvsd –**a** | *vsd_name* ...

Flags

–**a** Specifies that all IBM Virtual Shared Disks in the stopped state are to be unconfigured.

Operands

vsd_name Specifies an IBM Virtual Shared Disk.

Description

The **ucfgvsd** command unconfigures the specified IBM Virtual Shared Disks. This command does not change any IBM Virtual Shared Disk definitions. It moves IBM Virtual Shared Disks from the stopped state to the defined state.

If a configured HSD is using this IBM Virtual Shared Disk, you must first unconfigure the HSD before you unconfigure the IBM Virtual Shared Disk.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Unconfigure an IBM Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/ucfgvsd Specifies the command file.

Security

You must have root privilege to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **suspendvsd**

Examples

To unconfigure the IBM Virtual Shared Disk **vsd1vg1n1** in the stopped state, enter:

```
ucfgvsd vsd1vg1n1
```

unallnimres

Purpose

unallnimres – Deallocates Network Installation Management (NIM) resources from a NIM master to one or more NIM clients.

Syntax

unallnimres –**h** | –**l** *node_list*

Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**l** *node_list* Indicates by *node_list* the SP nodes to which to unallocate installation resources. The *node_list* is a comma-separated list of node numbers.

Operands

None.

Description

Use this command to unallocate all NIM resources from a NIM client.

Standard Error

This command writes error messages (as necessary) to standard error.

Exit Values

0 Indicates the successful completion of the command.

–**1** Indicates that an error occurred.

Security

You must have root privilege to run this command.

Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

Location

/usr/lpp/ssp/bin/unallnimres

Related Information

Commands: **allnimres**, **setup_server**

Examples

To unallocate boot/installation resources to boot/install client nodes 1, 3, and 5 from their respective boot/install servers, enter:

```
unallnimres -l 1,3,5
```

undefhsd

Purpose

undefhsd – Undefines a data striping device (HSD).

Syntax

undefhsd *hsd_name*...

Flags

None.

Operands

hsd_name Specifies the unique name defined in the SDR names that you want to delete.

Description

This command is used to remove a data striping device (HSD) by removing its definition from the system, including the special device files in /**dev**. The HSDs must be unconfigured and in the defined state on all nodes in the system partition.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Undefine a Hashed Shared Disk option.

Files

/usr/lpp/csd/bin/undefhsd Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defhsd**, **hsdatalst**

Examples

To delete the HSD information associated with the data striping device *hsd1* from the SDR, enter:

```
undefhsd hsd1
```

undefvsd

Purpose

undefvsd – Undefines an IBM Virtual Shared Disk.

Syntax

undefvsd *vsd_name* ...

Flags

None.

Operands

vsd_name Specifies the IBM Virtual Shared Disk whose underlying logical volume you no longer wish to be globally accessed by any IBM Virtual Shared Disk nodes.

Description

This command is used to remove IBM Virtual Shared Disk definition data from the System Data Repository (SDR) and any special device files from /**dev** for the given *vsd_names* on all the IBM Virtual Shared Disk nodes. The IBM Virtual Shared Disks must be unconfigured and in the defined state on all the IBM Virtual Shared Disk nodes.

You can use the System Management Interface Tool (SMIT) to run the **undefvsd** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Undefine a Virtual Shared Disk option.

Files

/usr/lpp/csd/bin/undefvsd Specifies the command file.

Security

You must be in the **bin** group to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Command: **defvsd**

Examples

To delete the IBM Virtual Shared Disk information associated with IBM Virtual Shared Disk **vsd1vg2n1** from the SDR, enter:

```
undefvsd vsd1vg2n1
```

updatehsd

Purpose

updatehsd – Lets you change the option in the System Data Repository (SDR) that prevents overwriting the Logical Volume Control Block (LVCB) for specified data striping devices (HSDs).

Syntax

updatehsd {–**d** *hsd_names* | –**a}** [–**f**]
–**o** {**protect_lvcb** | **not_protect_lvcb**}

Flags

–**d** Specifies the names of the HSDs that are the targets of this command.

–**a** Updates the option on all HSDs defined in the system or system partition.

–**o protect_lvcb | not_protect_lvcb**
Specifies whether to skip the first stripe (the LVCB) up to a maximum of 128KB of every IBM Virtual Shared Disk that constitutes the HSD. **protect_lvcb** specifies skipping the LVCB; **not_protect_lvcb** specifies not skipping it.

–**f** Forces the SDR changes by reconfiguring one or more IBM Virtual Shared Disks on all nodes in the current partition on which those IBM Virtual Shared Disks are currently configured.

Operands

None.

Description

Use this command only on the control workstation.

Note: This utility is very powerful. Misuse can destroy the contents of a database. Only the superuser should be allowed to run it. If a database has been loaded on a configured HSD, modifying the *protect_lvcb* or *not_protect_lvcb* option will destroy the database.

The HSD name must be specified. You must choose either **protect_lvcb** or **not_protect_lvcb**. The data striping device must be defined in the System Data Repository (SDR).

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit set_HSDdd_parms
```

and select the Update Hashed Shared Disk Options. option or

```
smit hsd_mgmtd_parms
```

and select the Set/Show HSD Device Driver Operational Parameters. option or the Update Hashed Shared Disk Options option.

Files

/usr/lpp/csd/bin/updatehsd
Specifies the file that contains the command.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defhsd**, **dshsd**, **hsdatalst**

Examples

To set the **protect_lvcb** option for hsdcont01 and hsdcont02, enter:

```
updatehsd -d hsdcont01,hsdcont02 -o protect_lvcb
```

updatevsdnode

Purpose

updatevsdnode – Changes IBM Virtual Shared Disk options in the System Data Repository (SDR).

Syntax

updatevsdnode {–**n** *node_number* | **ALL** } {[–**a** *VSD_adapter*]
[–**i** *init_cache_buffer_count*]
[–**m** *max_cache_buffer_count*] [–**r** *vsd_request_count*]
[–**p** *rw_request_count*] [–**b** *min_buddy_buffer_size*]
[–**x** *max_buddy_buffer_size*] [–**s** *max_buddy_buffers*]
[–**?** *vsd_max_ip_packet_size*]}
[–**f**]

Flags

–**n** Specifies the node numbers of the nodes whose SDR information you want this command to update, or **ALL** nodes in the system or system partition. You can issue the command **/usr/lpp/ssp/install/bin/node_number** to find out the node number of the node you are running on.

–**a** Specifies the adapter name to be used for IBM Virtual Shared Disk communications with this node or nodes. The High Performance Switch (adapter name **css0** is recommended for the IBM Virtual Shared Disk for best performance.

–**i** The IBM Virtual Shared Disk device driver implements an optional write-through cache of pinned kernel memory with a block size of 4KB. When the first cached IBM Virtual Shared Disk is configured on a node, the cache is created, and it contains the number of blocks specified in this field of the SDR. The minimum value is 1. The recommended value is 64, which results in a 256K cache. If you use the switch as the IBM Virtual Shared Disk adapter, no cache buffer is allocated.

–**m** The number of buffers in the cache can be increased up to *max_cache_buffer_count*. You cannot decrease the number; you must unconfigure all the IBM Virtual Shared Disks and start over. The recommended value is 256, which results in a 1MB cache. If you use the switch as the IBM Virtual Shared Disk adapter, no cache buffer is allocated.

–**r** Specifies the maximum number of outstanding IBM Virtual Shared Disk requests originating on each node. If the number is too small, local requests will queue up waiting for a request block to become available. The recommended value is 256. The size of the block is approximately 76 bytes.

–**p** Specifies the maximum number of outstanding read/write requests (*rw_pbuf_count*) the IBM Virtual Shared Disk will make to each underlying logical volume. A pbuf is a little bigger than a buf structure, which is approximately 104 bytes. The minimum value is 1. The recommended value for a server node is 48. See the chapter

that describes IBM Virtual Shared Disks in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

–b Specifies the smallest buddy buffer a server uses to satisfy a remote request to an IBM Virtual Shared Disk. This value must be a power of 2 and greater than or equal to 4096. The recommended value is 4096 (4KB).

–x The largest buddy buffer a server will use to satisfy a remote request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. The maximum value, and the recommended value, is 65536 (64KB). This value must be the same on all nodes within a system partition.

–s The size of the buddy buffer affects the number of remote requests the IBM Virtual Shared Disk server node can handle at one time. Remote requests can queue waiting for a buddy buffer. **statvsd** reports this queuing as buddy buffer shortages. Use the output from **statvsd** to select a buddy buffer size for your environment. The recommended value when the switch is used as the IBM Virtual Shared Disk adapter, is at least 4, which results in a 256KB combined buddy buffer size in combination with the default *max_buddy_buffer_size* of 64KB.

–? The maximum IP message size for IBM Virtual Shared Disks, in bytes. The default value is 24KB (24,576). If you are using the switch as your IBM Virtual Shared Disk adapter, use a value of 60KB (61,440).

–f Specifies that this command will force the SDR changes by reconfiguring one or more IBM Virtual Shared Disks on all nodes in the current partition on which those IBM Virtual Shared Disks are currently configured.

Operands

None.

Description

Use **updatevsdnode** to change the specified values in the SDR for all nodes in *node_list*.

Note: This command only changes the information in the SDR. In order to effectively configure the IBM Virtual Shared Disks, you must first unconfigure all the IBM Virtual Shared Disks and then reconfigure them.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Set/Show IBM Virtual Shared Disk Device Driver Operational Parameters option or the Update IBM Virtual Shared Disk Device Driver Node Parameters option.

Files

/usr/lpp/csd/bin/updatevsdnode
Specifies the command file.

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Command: **vsdatalst**, **vsdnode**

Examples

To change buddy buffer options after you've installed a new SP Switch Adapter, enter:

```
updatevsdnode ALL -b 4096 -x 4 -s 5
```

This command leaves *min_buddy_buffer_size* at 4KB, the default, lowers the *max_buddy_buffer_size* to 4KB as well, and allocates a maximum of 5 buddy buffers.

updatevsdtab

Purpose

updatevsdtab – Changes the IBM Virtual Shared Disk **cache**/**nocache** option in the System Data Repository (SDR).

Syntax

updatevsdtab {–**v** *vsd_names* | –**a**} –**o cache** | **nocache** [–**f**]

Flags

–**v** *vsd_names*
: Specifies a list of IBM Virtual Shared Disk names to be updated.

–**a**
: Specifies that the option is to be changed on all nodes of the system or system partition.

–**o cache** | **nocache**
: Specifies either the **cache** or the **nocache** option. The default is **cache**.

–**f**
: Forces SDR changes by reconfiguring an IBM Virtual Shared Disk on all nodes in the current system partition on which the IBM Virtual Shared Disk is configured.

Operands

None.

Description

Use this command to update the SDR, if necessary. If the –**f** flag is specified, the IBM Virtual Shared Disks involved will be reconfigured (using **sysctl**) on all nodes that are up and initially had these IBM Virtual Shared Disks configured.

Note: The **defvsd** command allows you to rename an IBM Virtual Shared Disk and update the **cache** option for that IBM Virtual Shared Disk.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Set/Show IBM Virtual Shared Disk Device Driver Operational Parameters option or the Update IBM Virtual Shared Disk Options option.

Files

/usr/lpp/csd/bin/updatevsdtab

Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defvsd**, **updatevsdnode**

Examples

To change the cache default for all IBM Virtual Shared Disks on a system or system partition, type:

```
updatevsdtab -a -o nocache
```

verparvsd

Purpose

verparvsd – Verifies IBM Virtual Shared Disk system partitioning.

Syntax

verparvsd [–**F**] [–**o** *output_file*] *layout_directory* [*new_partition* ...]

Flags

–**F** Returns success if correctable IBM Virtual Shared Disk errors are found in the system partitioning operation. This flag is the same as **spapply_config** –**F** and is used only when **spapply_config** invokes **verparvsd** when it is invoked with –**F**.

–**o** Specifies the file where the System Data Repository (SDR) commands are placed to load the IBM Virtual Shared Disk data in the new system partitions. If –**o** is not specified, the output is placed in the /**spdata**/**sys1**/**vsd**/**partitionVSDdata** file.

Operands

layout_directory
Specifies the *layout_directory* that describes the new system partitions that the user wants to apply, and wants **verparvsd** to verify for IBM Virtual Shared Disk system partitioning. This operand is used as the first argument in the invocation of the **spdisplay_config** command. Refer to the **spdisplay_config** command for more details.

new_partition **...**
Specifies the list of new system partitions to be processed. If some system partitions are going to be unaffected by the system partitioning operation implied by the *layout_directory*, and you do not want **verparvsd** to look at them, do not list them here, but list only the system partitions being affected. The **verparvsd** command only verifies and processes the system partitions passed as arguments. If no new system partitions are given as arguments, **all** system partitions in *layout_directory* are processed and analyzed. The **spapply_config** command invokes **verparvsd** listing only the new, changing system partitions.

Description

Use this command to verify that the system partition proposed in the *layout_directory* will work for all the existing IBM Virtual Shared Disk data. The **spapply_config** command invokes this command to partition the IBM Virtual Shared Disk data during a system partition operation. The **verparvsd** command extracts all IBM Virtual Shared Disk data from nodes involved in the system partitioning and writes SDR commands to the output file that will reload the IBM Virtual Shared Disk SDR data into the correct new system partitions. This file is executed during the system partitioning process to partition the IBM Virtual Shared Disk data.

The **spapply_config** command invokes this command and its output to effect IBM Virtual Shared Disk system partitioning. You can also invoke the command prior to invoking the **spapply_config** command to see how well suited the desired layout is for the existing IBM Virtual Shared Disk configuration as defined in the SDR.

This command only checks and processes the new system partitions listed on the command line. If some existing system partitions are to be unchanged in the system partitioning operation, do not list those system partition names on the command line. If no new system partitions are listed, the default is to process all system partitions in the layout directory.

This command checks to see if the IBM Virtual Shared Disk data can be partitioned as specified by the layout directory without any problems. The command reports any problems it identifies, as well as reports how it would *fix* the problem.

The **verparvsd** command places global volume groups (GVGs) in the system partition containing their primary server node. IBM Virtual Shared Disks are placed in the system partition of their GVG. HSDs are placed in the system partition containing their first IBM Virtual Shared Disk.

The **verparvsd** command looks for the following types of errors in each new system partition:

1. Inconsistent VSD_adapter Node attributes. If any are found, the VSD_adapter field is set to en0 for all IBM Virtual Shared Disk nodes in the system partition.
2. Inconsistent VSD_max_buddy_buffer_size Node attributes. The **verparvsd** command sets the VSD_max_buddy_buffer_size field for all IBM Virtual Shared Disk nodes in the system partition to the largest value of any node in the system partition, and adjusts the VSD_max_buddy_buffers so that the buddy buffer is still the same size, or just minimally larger than it was before on each node.
3. A twin-tailed GVG with primary and secondary server nodes in different system partitions. GVGs are placed in the system partition of the primary server. If the secondary is in a different system partition, the **verparvsd** command will set the secondary server to NULL, making the GVG have only one server, the primary.
4. An HSD with IBM Virtual Shared Disks in more than one system partition. The **verparvsd** command appends .BAD to the HSD's name. These HSDs would be unusable if the new system partition were applied and the VSD_adapter was **css0**.

   As a corollary, if an HSD with .BAD at the end of its name is found in the new system partition to have all its IBM Virtual Shared Disks in the system partition, the .BAD will be removed from its name.
5. Any duplicate GVG, IBM Virtual Shared Disk, or HSD name. The **verparvsd** command keeps the original name for the first name it encounters, but makes up unique names for any subsequent duplicate names encountered. New names follow the following recommended naming conventions:

   GVG vg01n01 for single tailed GVG on node 1.
   vg01p0ss02 for twin tailed GVGs, primary server node 1, secondary server node 2.
   VSD vsd01vg01n01 (for example, vsdnn*GVG name*)
   HSD hsd01 (for example, hsdnn)

Files

/spdata/sys1/vsd/partitionVSDdata
The default location of the output file containing all the SDR commands to correctly system partition the IBM Virtual Shared Disk data.

Exit Values

The **verparvsd** command looks for error types (described previously) in each new system partition and corrects them as specified:

- Without –**F**, any single error of the preceding type for even a single system partition causes **verparvsd** to return an error code upon completion.
- With –**F**, all errors of all of the types described previously are corrected as described, and **verparvsd** returns a zero (successful) return code.

In either case, **verparvsd** processes all the IBM Virtual Shared Disk data, and generates a complete list of errors on standard error, and a complete SDR command list to the output file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **defhsd**, **defvsd**, **spapply_config**, **spdisplay_config**, **vsdnode**, **vsdvg**

Examples

To see how well suited the configuration specified in the **config.4_4_8/layout.6** layout directory is to your IBM Virtual Shared Disk configuration, enter:

```
verparvsd config.4_4_8/layout.6
```

vhostname

Purpose

vhostname – Sets or displays the virtual host name.

Syntax

vhostname [–**s**] [*host_name*]

Flags

–**s** Trims any domain information from the printed name.

Operands

host_name Sets the virtual host name to *host_name*.

Note: You must have root authority to use the *host_name* operand.

Description

Use this command to display or set the virtual host name of the local host. Only users with root authority can set the virtual host name. The *host_name* is stored in the /**etc**/**vhostname** file.

If displaying the virtual host name and the virtual host name has not been set and the /**etc**/**vhostname** file does not exist, **vhostname** will return the real host name from the kernel variable.

When setting the virtual host name, if the /**etc**/**vhostname** file does not exist, it will be created. If it does exist, the file contents will be overwritten by the new virtual host name.

To clear the virtual host name so that the virtual host name no longer exists, remove the /**etc**/**vhostname** file.

Note: You must have root authority to remove the /**etc**/**vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may fail or act incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **vhostname** instead of **hostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the fail over node. It should be noted that changing the real host name in the kernel can cause problems with other applications that rely on the real host name in the kernel to identify the *physical machine*.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts provided with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP) set and clear the virtual host name in the HACMP pre- and post-event scripts. The administrator normally should not have to set or clear the virtual host name.

Files

/etc/vhostname
Contains the virtual host name.

Exit Values

0 Indicates that if a parameter was used, a virtual host name was successfully set. If a parameter was not used, either a virtual or real host name was printed out.

1 Indicates that an error occurred.

Related Information

Subroutines: **getvhostname**, **setvhostname**

AIX command: **hostname**

AIX Subroutines: **gethostname**, **sethostname**

Examples

1. To display the virtual host name, enter:

   ```
   vhostname
   ```

2. To set the virtual host name to **spcw_prim**, enter:

   ```
   vhostname spcw_prim
   ```

3. To display the virtual host name and trim domain information for host **donald.ibm.com**, enter:

   ```
   vhostname -s
   ```

   A vhostname of **donald** prints out.

4. To clear the virtual host name so it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   Note: You must have root authority to remove the /**etc**/**vhostname** file.

vsdatalst

Purpose

vsdatalst – Displays IBM Virtual Shared Disk system definition data from the System Data Repository (SDR).

Syntax

vsdatalst [–**G**] –**g** | –**n** | –**v**

Flags

–**G** Displays information for all system partitions on the SP, not only the current system partition.

Only one of the following flags can be specified with each invocation of **vsdatalst**:

–**g** Displays the following SDR IBM Virtual Shared Disk global volume group data:

global_group_name,
local_group_name,
primary_server_node,
secondary_server_node. (This is only enabled with IBM Recoverable Virtual Shared Disk)

–**n** Displays the following SDR IBM Virtual Shared Disk Node data:

node_number,
host_name,
adapter_name,
init_cache_buffer_count,
max_cache_buffer_count,
rw_request_count,
vsd_request_count,
min_buddy_buffer_size,
max_buddy_buffer_size,
max_buddy_buffers.

–**v** Displays the following SDR IBM Virtual Shared Disk Definition data:

vsd_name,
logical_volume_name,
global_group_name,
minor_number,
option (cache|nocache).

Operands

None.

Description

Use this command to display one of several kinds of information to standard output.

You can use the System Management Interface Tool (SMIT) to run the **vsdatalst** command. To use SMIT, enter:

```
smit list_vsd
```

and select the option for the kind of IBM Virtual Shared Disk SDR information you wish to see.

Security

You must be in the **bin** group to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Examples

1. To display SDR IBM Virtual Shared Disk global volume group data, enter:

   ```
   vsdatalst -g
   ```

   The system displays a message similar to the following:

   Note: *backup* or *secondary_server_node* is only enabled with IBM Recoverable Virtual Shared Disk.

   ```
   VSD Global Volume Group Information
                                                  Server Node Numbers:
   Global Volume Group name        Local VG name   primary      backup
   ------------------------------- --------------- -------      ------
   hunter-rileysvg                 rileysvg              1           0
   ppstest1-rootvg                 rootvg                3           0
   tattooine-rootvg                rootvg                2           0
   ```

2. To display SDR IBM Virtual Shared Disk Node data, enter:

   ```
   vsdatalst -n
   ```

   The system displays a message similar to the following:

   ```
   VSD Node Information
                        Initial Maximum  VSD    rw    Buddy Buffer:
   node           VSD    cache   cache    req.   req.  min.  max.    size: #
   #    host_name adapt. buffers buffers  count  count size  size    maxbufs
   ---- --------- ------ ------- -------  -----  ----- ---- ----------------
    1   hunter     tr0     64      256     256    48    4096  65536     4
    2   tattooine  tr0     64      256     256    48    4096  65536     4
    3   ppstest1   tr0     64      256     256    48    4096  65536     4
   ```

3. To display SDR IBM Virtual Shared Disk Definition data, enter:

   ```
   vsdatalst -v
   ```

   The system displays a message similar to the following:

vsdatalst

```
VSD Table

VSD name          logical volume  Global Volume Group     minor# option
----------------- --------------- ----------------------- ------ ------
vsd.rlv01         rlv01           hunter-rileysvg              2 cache
vsd.rlv02         rlv02           hunter-rileysvg              3 cache
vsd.vsd1          vsd1            tattooine-rootvg             1 nocache
vsd.vsdp1         vsd1            ppstest1-rootvg              4 nocache
```

vsddiag

Purpose

vsddiag – Displays information about the status of IBM Virtual Shared Disks.

Syntax

vsddiag

Flags

None.

Operands

None.

Description

This command displays information about IBM Virtual Shared Disks that can help you determine their status and collect information that helps IBM service representatives diagnose system problems.

Note: The **vsddiag** command can only be used when no IBM Virtual Shared Disk I/O is in progress.

Files

/usr/lpp/csd/bin/vsddiag Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdatalst**, **vsdsklst**

Examples

To display information about the IBM Virtual Shared Disks in your system or system partition, enter:

```
vsddiag
```

If all IBM Virtual Shared Disk's are created and configured correctly, the output is:

```
Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
end of vsdl1diag:checkvsdl1 program.
```

If there are no IBM Virtual Shared Disk's defined, the output is:

```
k5n02.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

k5n01.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
end of vsdl1diag:checkvsdl1 program.
```

If there is something wrong with the IBM Virtual Shared Disk's, the output is:

```
k5n02.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

k5n01.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
vsdl1diag:checkvsdl1: 0034-619 Device driver on node 14 is not at the
same level as others on this SP system or system partition.
vsdl1diag:checkvsdl1: 0034-620 VSD Maximum IP Message Size on node 14 is
not at the same level as others on this SP system or system partition.
```

vsdelnode

Purpose

vsdelnode – Removes IBM Virtual Shared Disk information for a node or series of nodes from the System Data Repository (SDR).

Syntax

vsdelnode *node_number* ...

Flags

None.

Operands

node_number Specifies the number attribute assigned to a node in the SDR.

Description

This command is used to remove IBM Virtual Shared Disk data for a node or series of nodes from the SDR.

The **vsdelnode** command makes the listed nodes no longer IBM Virtual Shared Disk nodes so that no IBM Virtual Shared Disks can be accessed from them. This command fails for any nodes that are servers for any global volume groups.

You can use the System Management Interface Tool (SMIT) to run the **vsdelnode** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Delete IBM Virtual Shared Disk Node Information option.

Security

You must be in the **bin** group to run this command.

Restrictions

If you have the IBM Recoverable Virtual Shared Disk product installed and operational, do not use this command. The results may be unpredictable.

See *IBM Parallel System Support Programs for AIX: Managing Shared Disks*.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdatalst**, **vsdnode**

Examples

To delete IBM Virtual Shared Disk node information for nodes **3** and **6**, enter:

```
vsdelnode 3 6
```

vsdelvg

Purpose

vsdelvg – Removes IBM Virtual Shared Disk global volume group information from the System Data Repository (SDR).

Syntax

vsdelvg [–**f**] *global_group_name* ...

Flags

–**f**	Forces the removal of any IBM Virtual Shared Disks defined on this global volume group.

Operands

global_group_name	Specifies the volume group that you no longer want to be global to the system.

Description

Use this command to remove IBM Virtual Shared Disk global volume group information from the SDR. If any IBM Virtual Shared Disks are defined on a global volume group, the **vsdelvg** command fails unless –**f** is specified. If –**f** is specified, any such IBM Virtual Shared Disks must be unconfigured and in the defined state on all the IBM Virtual Shared Disk nodes to be deleted.

You can use the System Management Interface Tool (SMIT) to run the **vsdelvg** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Delete IBM Virtual Shared Disk Global Volume Group Information option.

Files

/usr/lpp/csd/bin/vsdelvg	Specifies the command file.

Security

You must be in the **bin** group to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdatalst**, **vsdvg**, **undefvsd**

Examples

To delete the IBM Virtual Shared Disk information associated with global volume group **vg1n1** from the SDR, enter:

```
vsdelvg vg1n1
```

vsdnode

Purpose

vsdnode – Enters IBM Virtual Shared Disk information for a node or series of nodes into the System Data Repository (SDR).

Syntax

vsdnode *node_number... adapter_name init_cache_buffer_count max_cache_buffer_count vsd_request_count rw_request_count min_buddy_buffer_size max_buddy_buffer_size max_buddy_buffers vsd_max_ip_msg_size*

Flags

None.

Operands

node_number — Specifies the node or nodes whose IBM Virtual Shared Disk information is to be set as identified by the *node_number* attribute of the SDR node class.

adapter_name — Specifies the adapter name to be used for IBM Virtual Shared Disk communications for the nodes specified. The adapter name must already be defined to the nodes. Note that the nodes involved in IBM Virtual Shared Disk support must be fully connected so that proper communications can take place. Use **css0** to specify that the IBM Virtual Shared Disk device driver transmits data requests over the High Performance Switch or the SP Switch. The **css0** adapter will be used the next time the IBM Virtual Shared Disk device driver is loaded.

init_cache_buffer_count — Specifies the initial number of buffers used for IBM Virtual Shared Disk caching for the nodes specified.

max_cache_buffer_count — Specifies the maximum number of buffers to be used for IBM Virtual Shared Disk caching for the nodes specified.

vsd_request_count — Specifies the number of outstanding IBM Virtual Shared Disk requests for the nodes specified.

rw_request_count — Specifies the number of outstanding read and write requests the IBM Virtual Shared Disk issues at one time to each logical volume for the nodes specified.

min_buddy_buffer_size — Specifies the smallest buddy buffer a server uses to satisfy a remote request to an IBM Virtual Shared Disk. This value must be a power of 2 and less than or equal to 4096. The recommended value is 4096 (4KB). For a 512 byte request, 4KB is clearly overkill. However, recall that a buddy buffer is only used while a remote request is being served at the serving node – a short period of time.

max_buddy_buffer_size Specifies the largest buddy buffer a server uses to satisfy a remote noncached request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. The recommended value is 65536 (64KB). This value depends on the I/O request size of applications using the IBM Virtual Shared Disks and the network used by the IBM Virtual Shared Disk software.

max_buddy_buffers This value should be the same on all nodes in a system partition. The buddy buffer is pinned kernel memory allocated when the IBM Virtual Shared Disk device driver is loaded the first time an IBM Virtual Shared Disk is configured (and freed when the last IBM Virtual Shared Disk is unconfigured). The size of the buddy buffer affects the number of remote requests the IBM Virtual Shared Disk server node can handle at one time. Remote requests can queue waiting for a buddy buffer. **statvsd** reports this queuing as buddy buffer shortages. Use this number to select the buddy buffer size for your environment. The recommended value is 4 initially, resulting in a 256KB buddy buffer for the switch.

vsd_max_ip_msg_size Specifies the maximum message size in bytes for IBM Virtual Shared Disks. The default is 24,576 (24KB) and that is the recommended value for an IP adapter. If you use the switch (**css0**) as the IBM Virtual Shared Disk adapter, change this value to 61,440 (60K).

Description

Use this command to make the specified nodes IBM Virtual Shared Disk nodes and to assign their IBM Virtual Shared Disk operational parameters. The operational parameters are: adapter name, initial cache buffer count, maximum cache buffer count, r/w request count, IBM Virtual Shared Disk request count, and buddy buffer parameters. If this information is the same for all nodes, run this command once. If the information is different for the nodes, run this command once for each block of nodes that should have the same IBM Virtual Shared Disk information.

You can use the System Management Interface Tool (SMIT) to run the **vsdnode** command. To use SMIT, enter:

```
smit vsd_data
```

and select the IBM Virtual Shared Disk Node Information option.

Files

/usr/lpp/csd/bin/vsdnode Specifies the command file.

Security

You must be in the **bin** group to run this command.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdatalst**, **vsdelnode**

Refer to *IBM Parallel System Support Programs for AIX: Managing Shared Disks* for defining IBM Virtual Shared Disk information in the SDR.

Examples

The following example adds SDR information for a **css0** network and nodes 1 through 8 with **css0** being the adapter name used for IBM Virtual Shared Disk communications. Initially, this is accomplished with 64 cache buffers and with a maximum of 256 cache buffers to be used for IBM Virtual Shared Disk support, with allowance for 48 outstanding r/w requests per IBM Virtual Shared Disk and 256 outstanding IBM Virtual Shared Disk requests per node with nine 64KB *max_buddy_buffers* and a 4KB *min_buddy_buffer_size*.

```
vsdnode 1 2 3 4 5 6 7 8 css0 64 256 256 48 4096 65536 9 61440
```

vsdsklst

Purpose

vsdsklst – Produces output that shows you the disk resources used by the IBM Virtual Shared Disk subsystem across a system or system partition.

Syntax

vsdsklst {–**d** | –**v** | –**dv**} {[–**a**] | –**n** *node_list*]} [–**G**]

Flags

–**d** Displays disk utilization information about volume groups and the physical disks associated with them.

–**v** Displays disk utilization information about volume groups and the IBM Virtual Shared Disks associated with them.

–**dv** Displays disk utilization information about volume groups, physical disks, and IBM Virtual Shared Disks.

–**a** Displays specified information for all nodes in the system or system partition.

–**n** *node_list* Lists one or more node numbers for which information is to be displayed.

–**G** Displays global disk information (across system partitions).

Operands

None.

Description

Use this command to check disk utilization across a system or system partition.

Files

/usr/lpp/csd/bin/vsdisklist
Specifies the command file.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Command: **vsdatalst**

Examples

This command:

```
vsdsklst -dv -a
```

displays the following information on a system that has volume groups and IBM Virtual Shared Disks defined on nodes 1, 3, 5, 7, 10, and 12. Node 5 is temporarily inactive.

```
k7n12.ppd.pok.ibm.com
Node Number:12; Node Name:k7n12.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:315
        Physical Disk:hdisk0; Total:537; Free:315
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD8n12{lv1HsD8n12}; Size:2
        VSD Name:1HsD20n12{lv1HsD20n12}; Size:2

k7n01.ppd.pok.ibm.com
Node Number:1; Node Name:k7n01.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:210
        Physical Disk:hdisk0; Total:537; Free:210
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD1n1{lv1HsD1n1}; Size:2
        VSD Name:1HsD13n1{lv1HsD13n1}; Size:2


k7n05.ppd.pok.ibm.com
No response

k7n10.ppd.pok.ibm.com
Node Number:10; Node Name:k7n10.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:303
        Physical Disk:hdisk0; Total:537; Free:303
        VSD Name:vsdn10v1{lvn10v1}; Size:4
        VSD Name:vsdn10v2{lvn10v2}; Size:4
        VSD Name:vsdn10v3{lvn10v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD6n10{lv1HsD6n10}; Size:2
        VSD Name:1HsD18n10{lv1HsD18n10}; Size:2


k7n03.ppd.pok.ibm.com
Node Number:3; Node Name:k7n03.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:269
        Physical Disk:hdisk0; Total:537; Free:269
        VSD Name:vsdn03v1{lvn03v1}; Size:4
        VSD Name:vsdn03v2{lvn03v2}; Size:4
        VSD Name:vsdn03v3{lvn03v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD2n3{lv1HsD2n3}; Size:2
        VSD Name:1HsD14n3{lv1HsD14n3}; Size:2
```

```
k7n07.ppd.pok.ibm.com
Node Number:7; Node Name:k7n07.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:300
        Physical Disk:hdisk0; Total:537; Free:300
        VSD Name:vsdn07v1{lvn07v1}; Size:4
        VSD Name:vsdn07v2{lvn07v2}; Size:4
        VSD Name:vsdn07v3{lvn07v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD4n7{lv1HsD4n7}; Size:2
        VSD Name:1HsD16n7{lv1HsD16n7}; Size:2
```

To view the output for a specific node, type:

```
vsdsklst -n 7
```

The output is:

```
k7n07.ppd.pok.ibm.com
Node Number:7; Node Name:k7n07.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:300
        Physical Disk:hdisk0; Total:537; Free:300
        VSD Name:vsdn07v1{lvn07v1}; Size:4
        VSD Name:vsdn07v2{lvn07v2}; Size:4
        VSD Name:vsdn07v3{lvn07v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD4n7{lv1HsD4n7}; Size:2
        VSD Name:1HsD16n7{lv1HsD16n7}; Size:2
```

Instead of issuing this command directly, you should use the appropriate SMIT panels to view it in the best format. To view information about volume groups, type:

```
smit lsvg
```

To view information about logical volumes, type:

```
smit lslv
```

To view information about physical volumes, type:

```
smit lspv
```

vsdvg

Purpose

vsdvg – Defines an IBM Virtual Shared Disk global volume group.

Syntax

vsdvg [–**g** *global_group_name*] *local_group_name primary_server_node* [*secondary_server_node*]

Flags

–**g** *global_group_name* — Specifies a unique name for the new global volume group. This name must be unique across the system partition. It should be unique across the SP, to avoid any naming conflicts during future system partitioning operations. The suggested naming convention is **vg***xx***n***yy*, where *yy* is the node number, and *xx* uniquely numbers the volume groups on that node. If this is not specified, the local group name is used for the global name. The length of the name must be less than or equal to 31 characters.

Operands

local_group_name — Specifies the name of a volume group that you want to indicate as being used for IBM Virtual Shared Disk. This name is local to the host upon which it resides. The length of the name must be less than or equal to 15 characters.

primary_server_node — Specifies the primary server node on which the volume group resides. The length of the name must be less than or equal to 31 characters. This can be specified in four different ways:

- frame,slot
- node number
- host name
- IP address

secondary_server_node — Specifies the secondary server node on which the volume group resides. The length of the name must be less than or equal to 31 characters.

This can be specified in four different ways:

- frame,slot
- node number
- host name
- IP address

Note: This operand is used only by the IBM Recoverable Virtual Shared Disk product.

vsdvg

Description

Use this command to define volume groups for use by IBM Virtual Shared Disk support. This is done by specifying the local volume group name, the node on which it resides, and the name by which the volume group will be known throughout the cluster.

You can use the System Management Interface Tool (SMIT) to run the **vsdvg** command. To use SMIT, enter:

```
smit vsd_data
```

and select the IBM Virtual Shared Disk Global Volume Group Information option.

Files

/usr/lpp/csd/bin/vsdvg Specifies the command file.

Security

You must be in the **bin** group to run this command.

Restrictions

The *secondary_server_node* operand is used only by the IBM Recoverable Shared Disk product.

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Command: **vsdelvg**

Examples

1. The following example adds SDR information indicating that the volume group known as **vg2n17** on node **17** is available for global access and is known to the cluster as **vg2n17**. Node **17** is the primary and only server.

   ```
   vsdvg vg2n17 17
   ```

2. The following example with the IBM Recoverable Virtual Shared Disk adds SDR information indicating that the volume group known as **vg1p3s15** on nodes **3** and **15** is available for global access and is known to the cluster as **vg1p3s15**. **3** is the primary server node and **15** is the secondary server node.

   ```
   vsdvg vg1p3s15 3 15
   ```

vsdvts

Purpose

vsdvts – Verifies that the IBM Virtual Shared Disk software works.

Syntax

vsdvts [–**b** *block_size*] [–**n** *number_of_blocks*] *vsd_name* [*file*]

Flags

–**b** Specifies the *block_size* used on the read and write calls to the IBM Virtual Shared Disk. Because the IBM Virtual Shared Disk raw device is used, the block size must be a multiple of 512. The default block size is 4096.

–**n** Specifies the number of blocks of the file to read. The default is to read 1MB of data from the file, so 1MB divided by *block_size* is the default number of blocks. Specifying 0 means to read as many full blocks of data as there are in the file. If more blocks are specified than are in the file, only the number of full blocks that exist will be used.

Operands

vsd_name Specifies the IBM Virtual Shared Disk to be verified (for example, that will be written and read with the data from the file). The IBM Virtual Shared Disk should be in the active state. Ensure that the IBM Virtual Shared Disk is large enough to hold all the data you plan to write to it. An IBM Virtual Shared Disk on a logical volume with one physical system partition is large enough if all the **vsdvts** defaults are taken.

file Specifies the file to be written to the IBM Virtual Shared Disk to verify its operation. The data is then read from the IBM Virtual Shared Disk and compared to this file to ensure the IBM Virtual Shared Disk read and write operations are successful. The default file is /**unix**.

Description

Warning

Warning: Data on *vsd_name* and its underlying logical volume is overwritten and, therefore, destroyed. Use this command after you have defined an IBM Virtual Shared Disk (including its underlying logical volume), but **before** placing application data on it.

Use this command to verify that the *vsd_name* is in the active state and then to write the specified part of *file* to the raw *vsd_name* device, /**dev**/**rvsd_name**. This command reads the data back from the IBM Virtual Shared Disk, then compares it to *file*. If the data is the same, the test is successful and **vsdvts** succeeds. Otherwise, **vsdvts** fails. The **dd** command is used for all I/O operations.

Try **vsdvts** on both a server and client node (for example, on both the node with a logical volume and one without it).

Prerequisite Information

IBM Parallel System Support Programs for AIX: Managing Shared Disks

Related Information

Commands: **vsdnode**, **vsdvg**, **defvsd**, **cfgvsd**, **startvsd**, **dd**

The preceding commands are listed in their order of use.

Part 2. Technical Reference

This part of the book contains the SP files, subroutines, and other technical information. It also contains the Distributed Tape Management System (DTMS) Enablement function I/O control requests.

© Copyright IBM Corp. 1996

Chapter 2. SP Files and Other Technical Information

© Copyright IBM Corp. 1996

haemloadlist file

Purpose

haemloadlist File – Event Management configuration data that is to be loaded into the System Data Repository (SDR)

Description

IBM Parallel System Support Programs for AIX (PSSP) includes a number of resource monitors that use the Resource Monitor Application Programming Interface (RMAPI) to supply information about system resources to the Event Management subsystem. Information about the monitors and the resources is defined in the **haemloadlist** file and includes:

- The resource variables from which events may be generated
- The instance vectors for each resource variable
- The classes in which the resource variables are grouped
- The resource monitors that supply the variables.

Before the Event Management subsystem can use this information, it must first be loaded into the System Data Repository (SDR) and then compiled into a binary Event Management Configuration Database (EMCDB) file. The **haemloadcfg** command is used to load the data from the **haemloadlist** file into the SDR. The **haemcfg** command is then used to compile the data from the SDR into the binary EMCDB file. Both of these commands are issued automatically by the **haemctrl** command when it is used to start the Event Management subsystem on the control workstation.

For resource monitors other than those supplied by PSSP, IBM recommends that you create one or more separate files in load list format to contain their Event Management configuration data. You can specify the name of any file in load list format on the **haemloadcfg** command.

You can use SDR commands to work with objects in the Event Management classes directly. However, IBM recommends that you use load list files and the **haemloadcfg** command to work with objects in these classes.

You cannot use System Management Interface Tool (SMIT) panels to work with objects in these classes.

The Format of a Event Management Load List File

Source information for the EMCDB is kept in the several partitioned classes in the SDR, each of which has a set of associated attributes. The format of an Event Management load list file, which can be used as input to the **haemloadcfg** command, follows the structure of the Event Management data in the SDR.

This man page describes the format of the Event Management load list file. For detailed information about its content (the Event Management SDR classes, attributes, and their values), see the description of the EMCDB in *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

The **haemloadlist** file consists of stanzas, each of which describes an object in one of the Event Management SDR classes. The stanzas may appear in any order in the file.

Each stanza begins with the SDR class name to which the object belongs, followed by one or more lines that define the attributes of the object. The SDR class name must begin in column 1.

An attribute line consists of the attribute name followed by an = (equal sign) followed by a data field. The = (equal sign) may not be surrounded by blank spaces. The data field consists of a single value for the attribute. If the field contains blanks, it must be surrounded by double quotes (**"**). If the field contains a double quote character, it should be preceded by a backslash (**"**). If the field contains a backslash character, it should be preceded a second backslash (\\).

The data field must be on the same line as the attribute. The attribute lines for a stanza stop at either the start of a new stanza or the end-of-file character.

Comments may be present in the file. Any line in which the first nonwhite space character is a pound sign (**#**) is a comment. Blank lines are also considered comment lines and are ignored.

The EM_Resource_Variable Stanza

The **EM_Resource_Variable** SDR class contains one object for each resource variable defined in the database. Accordingly, there is one **EM_Resource_Variable** stanza for each resource variable that is defined.

Here is an example of a stanza that defines a resource variable of type Quantity:

```
EM_Resource_Variable
        rvName="IBM.PSSP.aixos.PagSp.totalsize"
        rvDescription="99"
        rvValue_type="Quantity"
        rvData_type="long"
        rvInitial_value="0"
        rvClass="IBM.PSSP.aixos.PagSp"
        rvPTX_name="PagSp/totalsize"
        rvLocator="NodeNum"
        rvDynamic_instance="0"
```

Here is an example of a stanza that defines a resource variable of type State:

```
EM_Resource_Variable
        rvName="IBM.PSSP.Prog.pcount"
        rvDescription="1009"
        rvValue_type="State"
        rvData_type="SBS"
        rvClass="IBM.PSSP.Prog"
        rvLocator="NodeNum"
        rvDynamic_instance="1"
```

The EM_Structured_Byte_String Stanza

The **EM_Structured_Byte_String** SDR class contains one object for each structured field that is defined in a structured byte string. Accordingly, there is one **EM_Structured_Byte_String** stanza for each structured field that is defined.

The **IBM.PSSP.Prog.pcount** resource variable is an SBS that has three structured fields. Here is an example of the stanzas that define the fields:

```
EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="CurPIDCount"
        sbsField_type="long"
        sbsField_SN="0"

EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="PrevPIDCount"
        sbsField_type="long"
        sbsField_SN="1"

EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="CurPIDList"
        sbsField_type="cstring"
        sbsField_SN="2"
```

The EM_Instance_Vector Stanza

The **EM_Instance_Vector** SDR class contains one object for each instance vector element that is defined for a resource and all of its resource variables. Accordingly, there is one **EM_Instance_Vector** stanza for each instance vector element that is defined.

Here are some examples of stanzas that define instance vectors. The instance vector for the resource named **IBM.PSSP.aixos.PagSp** contains only one instance vector element. The instance vector for the resource named **IBM.PSSP.Prog** contains three elements.

```
EM_Instance_Vector
        ivResource_name="IBM.PSSP.aixos.PagSp"
        ivElement_name="NodeNum"
        ivElement_description="701"

EM_Instance_Vector
        ivResource_name="IBM.PSSP.Prog"
        ivElement_name="UserName"
        ivElement_description="701"

EM_Instance_Vector
        ivResource_name="IBM.PSSP.Prog"
        ivElement_name="ProgName"
        ivElement_description="701"

EM_Instance_Vector
        ivResource_name="IBM.PSSP.Prog"
        ivElement_name="NodeNum"
        ivElement_description="701"
```

The EM_Resource_Class Stanza

The **EM_Resource_Class** SDR class contains one object for each resource variable class that is defined in the database. Accordingly, there is one **EM_Resource_Class** stanza for each resource variable class that is defined.

Here are two examples of stanzas that define resource classes:

```
EM_Resource_Class
        rcClass="IBM.PSSP.aixos.PagSp"
        rcResource_monitor="aixos"
        rcObservation_interval="30"
        rcReporting_interval="0"

EM_Resource_Class
        rcClass="IBM.PSSP.Prog"
        rcResource_monitor="IBM.PSSP.harmpd"
        rcObservation_interval="0"
        rcReporting_interval="0"
```

The EM_Resource_Monitor Stanza

The **EM_Resource_Monitor** SDR class contains one object for each resource monitor that is defined in the database. Accordingly, there is one **EM_Resource_Monitor** stanza for each resource monitor that is defined.

Here are some examples of stanzas that define resource monitors:

```
EM_Resource_Monitor
        rmName="aixos"
        rmMessage_file="PEM.cat"
        rmMessage_set="3"
        rmConnect_type="internal"

EM_Resource_Monitor
        rmName="IBM.PSSP.harmpd"
        rmPath="/usr/lpp/ssp/bin/haemRM/harmpd"
        rmMessage_file="PEM.cat"
        rmMessage_set="2"
        rmConnect_type="server"
        rmPTX_prefix="0"
        rmPTX_description="0"
        rmPTX_asnno="0"

EM_Resource_Monitor
        rmName="IBM.PSSP.harmld"
        rmPath="/usr/lpp/ssp/bin/haemRM/harmld"
        rmMessage_file="harm_des.cat"
        rmMessage_set="1"
        rmConnect_type="server"
        rmPTX_prefix="IBM/PSSP.harmld"
        rmPTX_description="1,2"
        rmPTX_asnno="2"
```

Related Information

Commands: **haemloadcfg**, **haemcfg**

For a general overview of configuring Event Management, see “The Event Management Subsystem” chapter of *IBM Parallel System Support Programs for AIX: Administration Guide*.

haemloadlist File

For a description of the SDR classes and attributes that are related to the EMCDB, see *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*.

hmacls file

Purpose

/spdata/sys1/spmon/hmacls – Defines the Access Control Lists (ACLs) used by the Hardware Monitor daemon.

Description

The **/spdata/sys1/spmon/hmacls** file contains permission specifications for users to execute the various Hardware Monitor operations. Each line in the file consists of three white-space separated tokens in the following format:

```
obj     user_name      permissions
```

The *obj* token is either a frame ID or the host name of the control workstation (known as the Monitor and Control Node (MACN) by the Hardware Monitor). The *user_name* token is the user's principal name or principal name and instance.

The *permissions* token specifies which operations the user specified by the *user_name* token can execute against the object specified by the *obj* token.

If the *obj* token is a frame ID, the *permissions* token is one or more characters taken from the set **v**, **s**, and **m**. A definition of each follows:

v Specifies Virtual Front Operator Panel (VFOP) permission

s Specifies S1 permission

m Specifies Monitor permission

The VFOP permission implies the monitor permission. These permissions are described in the various commands. If the *obj* token is the MACN host name, the *permissions* token must be the character **a**. The character **a** is the administrative permission required to use the **hmadm** command.

Users are authorized to use the Hardware Monitor by virtue of having their names in the **/spdata/sys1/spmon/hmacls** file and have issued the **kinit** command with that name.

Files

/spdata/sys1/spmon/hmacls
Specifies ACLs for the Hardware Monitor daemon.

Related Information

Commands: **hmadm**, **hmcmds**, **hmmon**, **nodecond**, **s1term**

Examples

The following is an example of a **hmacls** file:

hmacls file

```
workstn3.kgn.ibm.com  john.admin  a
1  john.admin  vsm
2  john.admin  vsm
3  john.admin  vsm
1  mary        m
2  mary        m
3  mary        m
```

.klogin file

Purpose

.klogin – Specifies remote principals that can use a local user account.

Description

The **$HOME/.klogin** file defines which users and services on any remote hosts (computers in a network) within an authentication realm are allowed to invoke commands on the local user account.

The format of the **$HOME/.klogin** file is:

```
principal.instance@realm
```

A typical **.klogin** file, if present, looks similar to the following:

```
harry@KGN.IBM.COM
beverly.root@KGN.IBM.COM
root.admin@KGN.IBM.COM
user.wkst3@KGN.IBM.COM
user1@KGN.IBM.COM
rcmd.wkst3@KGN.IBM.COM
```

The **principal** name is the name of the remote user using SP authentication to execute remote commands on a local user account.

The **instance** is used to distinguish among variations on the principal name and may be used to indicate special privileges such as root authorization. In many environments, the **instance** is used with a service to indicate the workstation the service is running on. The service entries are usually found only in the root directory **.klogin** file.

The realm is the authentication realm. This may be different if there are several authentication realms, each with a different realm name, in your environment.

If the originating remote user is authenticated to one of the principals named in the **.klogin** file, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file. If a **.klogin** file is present, the owner must also be listed in order to gain access to his or her account from a remote host.

For security reasons, any **$HOME/.klogin** file must be owned by either the local user or root, and only the owner should have read and write access.

Files

$HOME/.klogin
Specifies remote users that can use a local user account.

Prerequisite Information

- Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.
- Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for an overview and for additional **Kerberos** information.

Related Information

SP Commands: **kinit**, **rcp**, **rsh**

SP Daemon: **kshd**

Examples

The following examples assume both Jeff and Anna have principal names in the authentication database (anna@KGN.IBM.COM, jeff@KGN.IBM.COM) and have issued a **kinit** to be authenticated on their local host. In addition, there is one authentication realm signified by KGN.IBM.COM.

1. To allow only user Anna on host wkst3 to rsh into her own account on host wkst7, a **.klogin** file is not required.
2. To allow user Jeff on host wkst3 to rsh into Anna's account on host wkst7, the **.klogin** file in Anna's account on wkst7 must have the following entries:

   ```
   anna@KGN.IBM.COM
   jeff@KGN.IBM.COM
   ```

   Anna's entry must be present in the **.klogin** file since the **.klogin** file exists. Jeff can now use the **–l** flag on the rsh or rcp to access Anna's account.

Kerberos

Purpose

Kerberos – Contains an introduction to SP authentication services.

Description

Kerberos authenticates individual users in a network environment. After authenticating your identity, you can use facilities such as Sysctl, the SP System Monitor, and the authenticated versions of network utilities **rsh** and **rcp**, without having to present passwords to remote hosts and without having to use **.rhosts** files.

Before you can use the SP authenticated commands, you must make sure that you are added to the authentication database. You can use the **kinit** command to find out. This command tries to authenticate your identity in the system. The **kinit** command prompts you for a principal name and password. Enter your principal name and password. If the command accepts your authentication information without sending you a message, you are already registered. If you enter your user name and **kinit** responds with the following message:

```
Kerberos principal unknown
```

contact your system administrator.

A principal identifier contains three parts. The first is the user or service name. The second is the instance, which in the case of a user is usually null. Some users can have privileged instances, however, such as root or admin. In the case of a service, the instance is the name of the machine on which it runs (for example, there can be a **rcmd** (**kshd** daemon) service running on the machine abc, which is different from the kshd service running on the machine xyz). The third part of the name is the realm. The realm corresponds to the service providing authentication for the principal. For example, computing resources within an enterprise can be partitioned into multiple administrative units for convenience or other business reasons.

When writing a principal identifier, the user or service name is separated from the instance (if not null) by a period, and the realm (if not the local realm) follows, preceded by an at sign (@).

When you authenticate your identify using the **kinit** command, you are given an initial ticket (which is an encrypted protocol message that provides authentication). This ticket is used to request other tickets from the authentication service for SP authenticated services such as SP **rsh** and SP **rcp**. The ticket transactions are done transparently, so you do not have to worry about their management.

Be aware that tickets expire. Some tickets, such as admin instance tickets, may expire in a few minutes, while other tickets may be good for hours, days, or weeks depending on the installation's policy. If your login session extends beyond the time limit, you have to reauthenticate your identity using the **kinit** command to get new tickets.

For more information about the **kinit** and **kdestroy** commands, see the **kinit** and **kdestroy** command.

Kerberos

Related Information

Commands: **kadmin**, **kdestroy**, **kinit**, **klist**, **kpasswd**

krb.conf file

Purpose

/**etc**/**krb.conf** – Contains the SP authentication configuration file.

Description

The **krb.conf** file contains configuration information describing the local authentication realm and the location of authentication servers for known realms.

The first line of the **krb.conf** file contains the name of the local authentication realm. Additional lines specify the location of an authentication server for a realm. The **krb.conf** file must contain at least one entry for each realm used by the local system. Each line specifying the location of an authentication server must be of the form:

```
REALM_NAME host_name
```

or

```
REALM_NAME host_name admin server
```

When “admin server” follows the host name, it indicates that the hosts also provides an administrative database server.

Related Information

SP File: **krb.realms**

Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional **Kerberos** information.

Examples

The following example of an /**etc**/**krb.conf** shows a simple configuration consisting a single realm with two servers, the primary and one secondary:

```
EAST.COAST
EAST.COAST master.authent.abc.com admin server
EAST.COAST backup.authent.abc.com
```

Here, “admin server” identifies the system whose full host name is “master.authent.abc.com” as the primary server, responsible for administration of the master database. Note that, in this case, there would have to be information in the /**etc**/**krb.realms** file to map the two host names or the domain name **authent.abc.com** to the local realm name, “EAST.COAST”. See the Example section of the **krb.realms** file.

krb.realms file

Purpose

/etc/krb.realms – Specifies the translations from host names to authentication realms.

Description

The **krb.realms** file provides a translation from a host name or a network domain name to the authentication realm name for the services provided by that host. Each line of the translation file is in one of the following forms (domain names should begin with a period (.)):

```
host_name    realm_name
domain_name  realm_name
```

If a host name exactly matches the *host_name* field in a line of the first form, the corresponding realm is the realm of the host. If a host name does not match any *host_name* in the file but its domain exactly matches a domain name, the corresponding realm is the realm of the host.

If no translation entry applies, the host's realm is considered to be the host name's domain portion converted to uppercase. If the host name does not contain a domain name, the host's realm is considered to be the host name converted to uppercase.

Related Information

SP File: **krb.conf**

Refer to the chapter on security in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional **Kerberos** information.

Examples

The following example of an **/etc/krb.realms** shows the entries that could be used to map a host name or a domain name to a realm. These names correspond to those used in the **krb.conf** file example.

```
master.authent.abc.com EAST.COAST
.authent.abc.com EAST.COAST
```

The first line maps a specific host name to the realm "EAST.COAST". If the host name were "master.east.coast", no entry would have been required. The second entry maps all host names whose domain portion is "authent.abc.com" to the same domain. The default mapping for this realm is:

```
.east.coast EAST.COAST
```

This type of mapping is always assumed, even if the **/etc/krb.realms** file is empty.

mt_tmcp

Purpose

mt_tmcp – Tape device driver (tape management enhancement) and tape management control point (TMCP)

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Description

Tape Device Driver (Tape Management Enhancement)

The tape management enhancement logic in the tape device driver (TDD) provides a set of functions that can be used by a management application to facilitate in the control of devices under the domain of a tape management system. This functionality provides a separate system call interface for tape manager access and a separate system call interface for end user access for the same physical tape device. The system call interface provided for tape manager access is referred to as the Management Protocol Interface (MPI). The system call interface provided for end user access is referred to as the User Protocol Interface (UPI).

The Management Protocol Interface (MPI) can be used to send tape position information, error and exception conditions, and operating mode information. It is also used by a tape management application to perform tape data access and control operations.

The User Protocol Interface (UPI) is the interface used for tape media access by the normal tape user for all read, write, and control operations.

Management Protocol Interface (MPI)

A Management Protocol Interface device is a device that was opened using a device special file that contains a major/minor number that has the (defined) MPI bit set. Refer to the section on TMCP and to the **mt_gdci** command for information on obtaining MPI major/minor numbers for tape devices configured to the tape device driver.

When a tape device is opened in MPI mode, the tape device driver recognizes that device as being under the domain of a tape management system and no longer accepts an **open()** system call (for that device) using the normal (non-MPI/UPI) minor number definition to reference the device.

A tape device only allows one MPI **open()** process at a time and rejects any additional **open()** system calls with **errno** set to EBUSY. A second MPI **open()** request causes a *ddreason* of OPEN2 to be returned by a set/read state information (SRSI) **ioctl()** issued to or sleeping on the device opened in Management Protocol Interface mode.

A device opened in Management Protocol Interface mode supports the standard AIX system calls of **read()**, **write()**, and **close()**. Support is also provided for those ioctls supported by the standard (non-MPI) tape device, as well as support for additional ioctls designated for use by devices opened in MPI mode.

A tape management system can use the MPI ioctls to pass operating mode information and control information to the TDD. It can also use the MPI device to perform tape positioning functions and to perform label processing on behalf of, and transparent to, the user of the device or volume (operating mode dependent).

An MPI **ioctl()** is used to inform a tape management system when a condition requiring external processing is detected by the tape device driver.

This interface is used to communicate operational information and error and exception information from the TDD to the tape manager. When a condition that requires tape manager notification occurs, the TDD suspends all activity on the affected device for the end user request and notifies the tape manager. The TDD uses the SRSI **ioctl()** system call to communicate with the tape manager. When an SRSI **ioctl()** system call is outstanding for a device that requires tape management action, the TDD indicates the reason for notification in the *ddreason* field of the request and returns the request to the tape manager. When an SRSI **ioctl()** system call is not outstanding for a device that requires tape management action, the TDD queues the notification indication until an SRSI **ioctl()** system call is issued for the device. The UPI end user process, when suspended awaiting tape manager action, remains suspended until direction is received from the tape manager through use of one of the *ddaction* subcommands of the SRSI **ioctl()** system call.

A tape management system can be notified (through this interface mechanism) to handle Header, Trailer, and User Label processing as dictated by the operating mode the user is running in and the tape position information as maintained by the TDD. A tape management system can also be notified (through this interface mechanism) to handle device state changes, post error processing, and close processing prior to the TDD returning to the user.

Refer to the **mt_srsi** command for information on the **ioctl()** used to set or change UPI logical device images state information in the physical tape device driver.

The following ioctls are provided to allow a tape management system to create and control UPI logical device image devices for use by end users access. These ioctls are only supported by a device that was opened in MPI mode:

CUPI Creates the User Protocol Interface (logical device image)
RUPI Removes the User Protocol Interface (logical device image)
SRMI Sets or reads mode information (logical device image)
SRSI Set or reads state information (logical device image)
SRSTI Sets or reads statistics information (logical device image)

The CUPI **ioctl()** system call creates a UPI logical device image for the addressed device. The **ioctl()** returns a set of UPI major/minor numbers for use by a tape manager to create the device special files to be used by an end user designated to use this UPI logical device image. Refer to the **mt_cupi** command for information on the **ioctl()** used to create UPI logical device images in the physical tape device driver.

The SRMI **ioctl()** system call sets or reads the operating mode of an existing UPI logical device image. Refer to the **mt_srmi** command for information on the **ioctl()** used to set or read UPI logical device images mode information in the physical tape device driver.

The RUPI **ioctl()** system call removes a previously created UPI logical device image. Refer to the **mt_rupi** command for information on the **ioctl()** used to remove UPI logical device images in the physical tape device driver.

The SRSI **ioctl()** system call sets or reads state information for an existing UPI logical device image. Refer to the **mt_srsi** command for information on the **ioctl()** used to set or read UPI logical device images state information in the physical tape device driver.

The SRSTI **ioctl()** system call sets or reads statistical information for a tape device. Refer to the **mt_srsti** command for information on the **ioctl()** used to set or read statistical information in the physical tape device driver.

When a **close()** system call is executed for an MPI device while the associated UPI logical device image still exists (was not removed by a RUPI **ioctl()**), a state of MPI forced close is indicated for this MPI device. The control block structure for the MPI device, as well as the control block structures established for the UPI logical device image remain intact until removed by a subsequent UNCONFIGURE operation that finds the device in the MPI forced close condition. In this case, all the associated control block structures for both the MPI and the UPI device are removed. A subsequent MPI **open()** system call for this device while in an MPI forced close condition is accepted and the MPI forced close condition is reset. All control block structures remain intact and subsequent UPI logical device image requests continue unchanged. Any UPI logical device image request that requires tape management action while the device is in an MPI forced close condition is returned to the user with an **errno** of ENXIO.

User Protocol Interface (UPI)

A user protocol interface device is a logical device image created by a tape management system to provide a secure path for end user access through an environment controlled by that tape manager. The user protocol interface device is created by a tape management system through use of the CUPI **ioctl()** system call issued to a device opened in Management Protocol Interface mode. Refer to the section on Management Protocol Interface mode described previously and the **mt_cupi** command for information on the creation of a UPI logical device image and the UPI major/minor numbers used to reference the device. A User Protocol Interface device can only be opened using a device special file that contains a major/minor number that has the (defined) UPI bit set.

The interface protocol support consists of the standard AIX system calls which include **open()**, **close()**, **read()**, and **write()**.

The **ioctl()** system call is also supported, but the functions provided through the **ioctl()** interface protocol vary depending on the operating mode allocated for a given user by the Tape Manager, as communicated through the Management Protocol Interface (MPI), and the support provided by the tape management system.

Tape Management Control Point (TMCP)

A tape management control point (TMCP) is a logical extraction of the tape device driver. It is provided by the TMS enhancement logic and can be used to extract configuration information from the tape device driver to provide notification relative to configuration change events within the tape device driver and to provide device driver program failure notification.

A TMCP device special file (/**dev**/**TMCP.major#**) is created automatically by the TMS enhancement logic when the first tape device is configured and is removed when the last tape device is unconfigured. A tape manager can use this TMCP entry to extract device configuration information from the tape device driver and to provide indirect access to MPI management **ioctl()** system calls that manipulate, initialize, and control logical device entries. A tape manager can also use the TMCP to receive configuration change information within the tape device driver, as well as exception condition reporting.

A TMCP only accepts an **open()** system call that contains the (defined) MPI bit set in the device minor number (MPI open request). Any **open()** system call issued to the TMCP without the MPI bit set in the minor number is rejected with **errno** set to EINVAL. A TMCP only allows one MPI **open()** process at a time and rejects any additional **open()** system calls with **errno** set to EBUSY. The second MPI **open()** also causes a *ddreason* of OPEN2 to be returned by a set/read state information (SRSI) **ioctl()** issued to or sleeping on the TMCP.

A TMCP uses the SRSI **ioctl()** system call to communicate with the tape manager. If an SRSI **ioctl()** system call is outstanding when an event occurs that requires tape management notification, the TMCP indicates the reason for notification in the *ddreason* field of the request and returns the request to the tape manager. If an SRSI **ioctl()** system call is not outstanding when an event occurs that requires tape management notification, the TMCP queues the notification indication until an SRSI **ioctl()** system call is issued for the TMCP.

A TMCP does not accept **read()** or **write()** system calls and rejects these requests with an **errno** value of EINVAL. A TMCP supports the following **ioctl()** requests:

CSK Creates the security key.

GDCI Gets device configuration information.

IOCINFO Gets general device information. Refer to the **tape** command for a list of related magnetic tape functions.

SRSI Set or reads state information. The only subcommands supported are RDMODE, SRSIWAIT, and WAITTIME, if the request is ultimately directed toward the TMCP.

With the exception of the other MPI **ioctl()** system call listed under the section on Management Protocol Interface (using the indirect addressing function of those ioctls), all other **ioctl()** system calls are rejected with **errno** set to EINVAL.

The GDCI **ioctl()** system call obtains tape device driver configuration information from a TMCP. The subcommands provided with this **ioctl()** allow the requestor to obtain all configuration entries from the device driver, or only those new entries or configuration change requests that were received. Each device configuration entry returned includes the major/minor number with the (defined) MPI bit set that is required to open that device as a Management Protocol Interface (MPI) device.

Refer to the **mt_gdci** command for information on the **ioctl()** used to get device configuration information from the physical tape device driver.

The SRSI **ioctl()** system call obtains event information from a TMCP. The PACTION subcommands are not supported when issued to a TMCP. Refer to the **mt_srsi** command for information on the **ioctl()** used to set or read UPI logical device images state information in the physical tape device driver.

The CSK **ioctl()** system call creates a security key for the TDD. This system call must be issued immediately following the **open()** system call issued to a TMCP. The security key established for the tape device driver must be included in the MPI **ioctl()** system calls on systems that do not support the “tmsmpi” C2 security privilege. Refer to the **mt_cskey** command for information on the **ioctl()** used to create a security key for the physical tape device driver.

Related Information

loctls: **mt_cskey**, **mt_cupi**, **mt_gdci**, **mt_rupi**, **mt_srmi**, **mt_srsi**, **mt_srsti**

Refer to the **mte_tmpcp** man page for information on the pseudo tape device driver and and the tape management pseudo control point (TMPCP).

mte_exits file

Purpose

mte_exits – Specifies the site-defined tape exit facility.

Description

The pseudo tape device driver supports a site-defined exit facility that can be invoked by a tape management system for user system calls (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) issued through the User Protocol Interface (UPI) of the pseudo tape device driver. These exits are provided as a site-defined option.

This exit capability allows individual site-specific processing to be invoked prior to device driver execution of each system call listed, and individual site-specific processing to be invoked prior to the return to the user at the completion of each system call listed.

This exit facility is implemented within the pseudo tape device driver with internal function call routines that can be replaced by site-supplied routines to provide the site-specific processing. The internal function call routine interfaces are defined later in this section.

A separate function call routine, known as a pre-call routine, is invoked for each (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) at the time the request is received.

A separate function call routine, known as a post-call routine, is also provided for each system call (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) that is invoked before returning to the user at the completion of the system call.

After system call entry to the pseudo tape device driver, the appropriate pre-call routine (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) is invoked for the UPI device addressed as defined by the device minor number supplied with the request. The pseudo tape device driver, upon return from the pre-call routine, invokes the tape device driver at the respective system call entry (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) for the UPI device requested using the device minor number supplied with the request.

Upon return from the tape device driver, the appropriate post-call routine (**open()**, **close()**, **read()**, **write()**, and **ioctl()**) is invoked for the UPI device addressed as defined by the device minor number supplied with the request.

Exit support is invoked for logical device image types of MAGTAPE and CP_REMOTE. Refer to the **mte_tmpcp** and the **mte_cupi** commands for information on creating UPI logical device images in the pseudo tape device driver.

The exit function call routines are defined in **/usr/lpp/mte/custom/mte_exits.c** and are described in the following sections. Refer to the **mte_make** file to compile and link the customized **mte_exits.c** into the pseudo tape device driver. The customized **mte_exits.c** must be rebased if a new version of **mte_exits.c** is shipped through IBM service. It must also be recompiled and relinked whenever the pseudo tape device driver is serviced or any supported tape device driver is serviced or shipped.

UPI open() Pre-call

This function call supports the site option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) **open()** system call.

int tm_uor (*devno***,***flag***,***pdev***)**
dev_t *devno*;
int *flag*,*pdev*;

Parameters

devno	Specifies the device major/minor number.
flag	Specifies the *Oflag* parameter passed by the user with the **open()** system call.
pdev	Specifies the physical device address assigned to this entry.

Return Value

A return value of 0 indicates successful completion; the pseudo tape device driver continues processing the call. A nonzero return value indicates an error; the pseudo tape device driver does not continue processing this call, but sets the return value in **errno**, and returns -1 to the user request.

UPI open() Post-call

This function call supports the option to perform specific site-supplied post processing of a User Protocol Interface (UPI) **open()** system call.

int tm_uoe (*devno***,***flag***,***ddret***)**
dev_t *devno*;
int *flag*;
int *ddret*;

Parameters

devno	Specifies the device major/minor number.
flag	Specifies the *Oflag* parameter passed by the user with the **open()** system call.
ddret	Specifies the device driver **int**/**errno** value that is returned to the user **open()**.

Return Value

The return value is ignored.

UPI close() Pre-call

This function call supports the option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) **close()** system call.

void tm_ucr (*devno***)**
dev_t *devno*;

Parameters

devno	Specifies the device major/minor number.

Return Value

None. The pseudo tape device driver continues processing the system call.

UPI close() Post-call

This function call supports the option to perform specific site-supplied post processing of a User Protocol Interface (UPI) **close()** system call.

int tm_uce (*devno***,***ddret***)**
dev_t *devno*
int *ddret***;**

Parameters

devno Specifies the device major/minor number.

ddret Specifies the device driver **int**/**errno** value that is returned to the user at **close()**.

Return Value

The **int**/**errno** value returned to the user. If this value is 0, a successful indication should be returned to the user. If the value is nonzero, it represents the **errno** to be sent to the user.

Note: The value passed in the *ddret* field to the site exit function should be passed as the return value if the site exit process does not want to alter the **int**/**errno** value being passed to the user by the tape device driver.

UPI read() Pre-call

This function call supports the site option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) **read()** system call.

int tm_urr (*devno***,***uiop***)**
dev_t *devno***;**
struct uio ***uiop;**

Parameters

devno Specifies the device major/minor number.

uiop Specifies the pointer to the UIO structure.

Return Value

A return value of 0 indicates successful completion; the pseudo tape device driver continues processing the call. A nonzero return value indicates an error; the pseudo tape device driver does not continue processing this call, but sets the return value in **errno**, and returns -1 to the user request.

UPI read() Post-call

This function call supports the site option to perform specific site-supplied post processing of a User Protocol Interface (UPI) **read()** system call.

int tm_ure (*devno***,***uiop***,***ddret***)**
dev_t *devno***;**
struct uio ***uiop;**

int *ddret*;

Parameters

devno Specifies the device major/minor number.

uiop Specifies the pointer to the UIO structure.

ddret Specifies the **errno** value (if nonzero) that was returned by the tape device driver.

Return Value

The **int**/**errno** value returned to the user. If this value is 0, a successful indication should be returned to the user. If the value is nonzero, it represents the **errno** to be sent to the user.

Note: The value passed in the *ddret* field to the site exit function should be passed as the return value if the site exit process does not want to alter the **int**/**errno** value being passed to the user by the tape device driver.

UPI write() Pre-call

This function call supports the site option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) **write()** system call.

int tm_uwr (*devno*,*uiop***)**
dev_t *devno*;
struct uio **uiop*;

Parameters

devno Specifies the device major/minor number.

uiop Specifies the pointer to the UIO structure passed with the **dd_write()** system call.

Return Value

A return value of 0 indicates successful completion; the pseudo tape device driver continues processing the call. A nonzero return value indicates an error; the pseudo tape device driver does not continue processing this call, but sets the return value in **errno**, and returns -1 to the user request.

UPI write() Post-call

This function call supports the site option to perform specific site-supplied post processing of a User Protocol Interface (UPI) **write()** system call.

int tm_uwe (*devno*,*uiop*,*ddret***)**
dev_t *devno*;
struct uio **uiop*;
int *ddret*;

Parameters

devno Specifies the device major/minor number.

uiop Specifies the pointer to the UIO structure.

ddret Specifies the **errno** value (if nonzero) that was returned by the tape device driver.

Return Value

The **int**/**errno** value returned to the user. If this value is 0, a successful indication should be returned to the user. If the value is nonzero, it represents the **errno** to be sent to the user.

Note: The value passed in the *ddret* field to the site exit function should be passed as the return value if the site exit process does not want to alter the **int**/**errno** value being passed to the user by the tape device driver.

UPI ioctl() Pre-call

This function call supports the site option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) **ioctl()** system call.

int tm_uir (*devno***,***req***,***arg_p***)**
dev_t *devno*;
int *req*;
caddr_t *arg_p*;

Parameters

devno Specifies the device major/minor number.

req Specifies the command parameter passed by the user with the **ioctl()**.

arg_p Specifies the *arg_p* parameter passed by the user with the **ioctl()**.

Return Value

A return value of 0 indicates that the pseudo tape device driver continues processing the system call. A return value greater than 0 represents an **errno** value to be returned to the user and, therefore, the pseudo tape device driver does not continue processing this system call, but passes this value with the system call return. The return value passed is placed in **errno** and -1 is returned to the user.

UPI ioctl() Post-call

This function call supports the site option to perform specific site-supplied post processing of a User Protocol Interface (UPI) **ioctl()** system call.

int tm_uie (*devno***,***req***,***arg_p***,***ddret***)**
dev_t *devno*;
int *req*;
caddr_t *arg_p*;
int *ddret*;

Parameters

devno Specifies the device major/minor number.

req Specifies the command parameter passed by the user with the **ioctl()**.

arg_p Specifies the *arg_p* parameter passed by the user with the **ioctl()**.

ddret Specifies the device driver **int**/**errno** value that was returned by the tape device driver.

Return Value

The **int**/**errno** value returned to the user. If this value is 0, the operation was successful. If this value is nonzero, it represents the **errno** value to be returned to the user.

Note: The value passed in the *ddret* field to the site exit function should be passed as the return value if the site exit process does not want to alter the **int**/**errno** value being passed to the user by the tape device driver.

UPI Device Change

This function call supports the site option to perform specific site-supplied preprocessing of a User Protocol Interface (UPI) device or type change.

void tm_udc(*old_devno***,** *new_devno***,** *old_pdev***,** *new_pdev***,** *old_type***,** *new_type***)**
dev_t *old_devno***,** *new_devno***;**
int *old_pdev***,** *new_pdev***;**
int *old_type***,** *new_type***;**

Parameters

old_devno Specifies the old device major/minor number.

new_devno Specifies the new device major/minor number.

old_pdev Specifies the old physical device address assigned to this entry.

new_pdev Specifies the new physical device address assigned to this entry.

old_type Specifies the old device type.

new_type Specifies the new device type.

Return Value

None.

Related Information

Command: **mte_tmpcp**

File: **mte_make**

mte_itable file

Purpose

mte_itable – Customizes the pseudo tape device driver to support new ioctls for device driver development.

Description

The pseudo tape device driver can be customized to support new ioctls for device driver development. The ioctl repository **/usr/lpp/mte/custom/mte_itable.c** includes all ioctls that can be issued against an MTE logical device image (see the **mte_cupi** command).

To add a new ioctl to the **mte_itable.c** repository, add or update a **UPI_ioctl** structure to the end of the **mteioctl_table[ ]** using the format described in **/usr/lpp/mte/custom/mte_ioctl.h**. It must also be recompiled and relinked whenever the pseudo tape device driver is serviced or a service update is installed that changes supported tape device driver header files.

Refer to the **mte_make** file to compile and link the customized **mte_itable.c** into the pseudo tape device driver. The customized **mte_itable.c** must be rebased if a new version of **mte_itable.c** is shipped through IBM service. It must also be recompiled and relinked whenever the pseudo tape device driver is serviced.

Note: All nodes that require remote tape access or nodes that have a physical tape driver attached that DTMS uses, must synchronize to use the customized **mte_itable.c** definition.

Related Information

Command: **mte_tmpcp**

File: **mte_make**

mte_make file

Purpose

mte_make – Contains the makefile for rebuilding the pseudo tape device driver.

Description

The distributed tape management system enablement function supports a site-defined exit facility (see the **mte_exits** file) and an ioctl repository (see the **mte_itable** file) for customization and integration into the pseudo tape device driver.

The /**usr**/**lpp**/**mte**/**custom** directory includes the following files for site customization:

Atape.h
mte.exp
mte_ioctl.h
mte_itable.c
mte_make
mte_exits.c
mtextend.h
mtio.h
mttms.h
obj/
sockets.exp

Note: Both mtextend.h and mtio.h are shipped with the AIX Parallel and ESCON Channel Tape Attachment/6000 device driver (mtdd). Refer to the *AIX Parallel and ESCON Channel Tape Attachment/6000 Installation and User's Guide* for additional information on the mtdd device driver. Refer to the *IBM SCSI Tape Drive and Library Device Drivers Installation and User's Guide* for additional information on the Atape device driver.

mte_make is the makefile for rebuilding the pseudo tape device driver. Header and export files are provided only for compilation and should not be modified. **mte_exits.c** and **mte_itable.c** can be customized as described in the **mte_exits** and **mte_itable** files, respectively. The obj directory contains the object files for the linkage editor and is the target directory in **mte_make** for the pseudo tape device driver.

To customize the pseudo tape device driver:

1. Copy /**usr**/**lpp**/**mte**/**custom** to a working directory.
2. Modify **mte_exits.c** and **mte_itable.c** as needed.
3. Issue the following command:

   ```
   make -f mte_make
   ```

4. Save /**usr**/**lpp**/**mte**/**mtedd** (see the Note that follows).
5. Copy the customized **obj**/**mtedd** to /**usr**/**lpp**/**mte**.
6. Deconfigure the TMPCPs to unload the current MTE pseudo device driver and then reconfigure the TMPCPs to load the customized pseudo tape device driver (see the “Distributed Tape Management System Enablement” chapter of *IBM*

Parallel System Support Programs for AIX: Administration Guide for configuration information).

Note: Copy the saved pseudo tape device driver from step 4 to **/usr/lpp/mte/mtedd** and repeat step 6 to restore the previous driver. To restore the default **mte_exits.c** and **mte_itable.c**, deinstall and reinstall the MTE **installp** image.

Related Information

Command: **mte_exits**, **mte_itable**

mte_tmpcp

Purpose

mte_tmpcp – Pseudo tape device driver and tape management pseudo control point (TMPCP).

Description

Magnetic Tape Extension (MTE) Pseudo Device Driver

The magnetic tape extension (MTE) pseudo tape device driver provides enabling facilities that can be used to support a distributed, multinode tape management system. Support is provided to create and control local and local/remote pair logical device images. The support structure is also provided for the invocation of site-defined exit processing for AIX system calls to logical (tape) devices. Refer to the following section for a description of the tape management pseudo control point (TMPCP). Refer to the **mte_exits** file for an overview and interface definition of the site exit facility provided by the pseudo tape device driver.

Tape Management Pseudo Control Point (TMPCP)

A tape management pseudo control point (TMPCP) is a logical extraction of the pseudo tape device driver that is used to create and control User Protocol Interface (UPI) logical device images. Each (configured) TMPCP is capable of supporting 255 logical device images. One device special file is created for each configured TMPCP for the pseudo tape device driver.

A TMPCP only accepts an **open()** that contains the MPI bit set in the device minor number (MPI **open()** request). Any **open()** issued to the TMPCP without the MPI bit set in the device minor number is rejected with an **errno** value of EINVAL. Each TMPCP only allows one **open()** process at a time and rejects additional **open()** requests with an **errno** value of EBUSY.

A TMPCP does not accept **read()** or **write()** system calls and rejects these requests with an **errno** value of EINVAL. Each TMPCP supports the following **ioctl()** system call requests:

MTTMSCSKE Creates a system security key.

MTTMSCUPIE Creates a User Protocol Interface (logical device image).

MTTMSIOCINFO Gets general device information.

MTTMSMUPIE Modifies a User Protocol Interface (logical device image).

MTTMSRUPIE Removes a User Protocol Interface (logical device image).

MTTMSSRSIE Sets and reads state information.

MTTMSEUPIE Inserts or extracts user data or **ioctl()** parameters (logical device image).

All other **ioctl()** requests are rejected with **errno** set to EINVAL.

The MTTMSCUPIE **ioctl()** system call request creates a UPI logical device image. The major/minor device number for the tape device that this logical device image is created for, is passed to the TMPCP in the *devmmn* field of the MTTMSCUPIE **ioctl()** structure. The MTTMSCUPIE **ioctl()** system call returns a set of UPI

major/minor numbers to be used by a tape manager to create the device special files for the user of this UPI logical device image. The UPI major/minor numbers are the mechanism that allows the pseudo tape device driver to be called for all references by the user to a UPI tape device. Refer to the "Distributed Tape Management System Enablement" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide* or to the **mt_tmcp** command for information on creating UPI logical device images in the physical tape device driver. Logical device image types of MAGTAPE, CP_LOCAL, CP_REMOTE, and STUB are supported by the MTE pseudo tape device driver. Refer to the **mte_cupi** command for information on the **ioctl()** used to create UPI logical device images in the pseudo tape device driver.

The MTTMSMUPIE **ioctl()** system call request changes or modifies an existing UPI logical device image. Refer to the **mte_mupi** command for information on modifying or changing UPI logical device images in the pseudo tape device driver.

The MTTMSRUPIE **ioctl()** system call request is used to remove a previously created logical device image from the pseudo tape device driver configuration. Refer to the **mte_rupi** command for information on removing UPI logical device images in the pseudo tape device driver.

The MTTMSCSKE **ioctl()** system call request is used to establish a security key in the pseudo tape device driver for tape management **ioctl()** access. Refer to the **mte_cskey** command for information on the **ioctl()** used to establish a security key in the pseudo tape device driver.

The MTTMSEUPIE **ioctl()** system call request is used for recovery operations to insert or extract user data or **ioctl()** parameters to or from a logical device image. Refer to the **mte_eupie** command for information on the **ioctl()** used for recovery operations.

A state of "MPI forced close" results when a **close()** system call is executed for the TMPCP and one or more logical device images created by a MTTMSCUPIE **ioctl()** still exist (not removed by a MTTMSRUPIE **ioctl()**). The control block structure for the TMPCP, as well as the control block structures established for each existing logical device image, remain intact until removed by a subsequent UNCONFIGURE operation that finds the TMPCP in the MPI forced-close condition, in which case all associated control block structures are removed. A subsequent **open()** system call for the TMPCP while in an MPI forced-close condition (not unconfigured) is accepted and the MPI forced-close condition is reset. All control block structures remain intact and subsequent logical device image requests continue unchanged.

Each UPI logical device image accepts **open()**, **close()**, **read()**, **write()**, and **ioctl()** system calls. For each system call received, the appropriate site-defined exit routine (optional) is called, and then the request is passed to the physical tape device driver on the local or remote node. Refer to the "Distributed Tape Management System Enablement" chapter of *IBM Parallel System Support Programs for AIX: Administration Guide* or to the **mt_tmcp** command for information about the tape management extensions in the physical tape device driver.

Related Information

Commands: **mte_cskey**, **mte_cupi**, **mte_eupi**, **mte_mupi**, **mte_rupi**, **mte_srsi**

Files: **mte_exit**, **mte_itable**

Refer to the **mt_tmcp** man page for information about the tape management extensions (MPI/UPI) and the tape management control point (TMCP) in the physical tape device driver.

SDR_dest_info file

Purpose

SDR_dest_info – Provides connection addresses for System Data Repository (SDR) clients.

Description

The **SDR_dest_info** file provides connection addresses for System Data Repository (SDR) clients. It contains the following fields:

primary Specifies the IP address of the system partition to which the node belongs. On the control workstation, this is the IP address of the default system partition.

default Specifies the IP address of the default system partition.

Related Information

Refer to the *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information on the **SDR_dest_info** file.

Examples

This example shows the contents of an **SDR_dest_info** file for a node. The node is a member of system partition 129.40.127.46. The default system partition on this system is 129.40.127.47.

```
default:129.40.127.47
primary:129.40.127.46
```

sysctl.acl file

Purpose

sysctl.acl – Contains the Access Control List (ACL) for the Sysctl ACL authorization callback.

Description

The **sysctl.acl** file contains the list of users authorized to access objects which are assigned the ACL authorization callback. Each line of the file must conform to one of the following formats:

Comment A line which begins with a number sign (#) is considered a comment.

Principal name A line of the form of "_PRINCIPAL *principal_name*" defines a principal on the ACL. If the *principal_name* is not fully qualified (for example, it does not contain an instance or realm separator), the realm defaults to the local realm and the instance defaults to NULL.

ACL file A line in the form "_ACL_FILE *file_name*" references another ACL file the contents of which is included in this file.

The first line of this (or any Sysctl ACL file) must begin with the line:

```
#acl#
```

Note: Including a principal in the **sysctl.acl** file gives that principal the same level of authority as having the root password.

Prerequisite Information

See the descriptions of SP authentication and Sysctl in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Related Information

SP Command: **sysctl**

SP Daemon: **sysctld**

Examples

The following **sysctl.acl** file defines two principals and includes the contents of another ACL file:

```
#acl#
#
# This is the Sysctl ACL file for the system.  It defines two
# principals: "root.@HPSSL.KGN.IBM.COM" and "rcr.@HPSSL.KGN.IBM.COM"
# It also includes the contents of file /etc/my.acl.file
#
_PRINCIPAL root.@HPSSL.KGN.IBM.COM
_PRINCIPAL rcr.@HPSSL.KGN.IBM.COM
_ACL_FILE /etc/my.acl.file
```

sysctl.conf file

Purpose

sysctl.conf – Configures the Sysctl server (**sysctld**) running on the local SP node.

Description

The **sysctl.conf** file configures the local Sysctl server daemon by optionally creating variables, procedures and classes, setting variables, loading shared libraries, and executing Sysctl commands. These items are used by the server in the processing of requests from local and remote Sysctl clients.

The default location of this file is /**etc**/**sysctl.conf**. An alternate configuration file location can be specified by using the –**f** flag when starting the server (see the **sysctld** command).

The /**etc**/**sysctl.conf** file contains Sysctl commands which are read and executed by the Sysctl server during its initialization. The following commands are supported:

create var *var_name var_value* **[***auth_callback***]**

The **create var** statement creates a variable, assigns it a value, and assigns it an authorization callback. This variable can then be referenced from within other Sysctl procedures and commands. If the *auth_callback* parameter is not supplied, a value of NONE is assumed. For example:

```
create var buildTop /usr/lpp/ssp AUTH
```

creates the variable **buildTop**, assigns it a value of /**usr**/**lpp**/**ssp**, and assigns it an authorization callback of AUTH. The variable **buildTop** can be referenced within Sysctl commands and procedures.

Another example:

```
create var STARTTIME [exec /bin/date] NONE
```

creates the variable **STARTTIME**, assigns it the value returned from the execution of the /**bin**/**date** command at server initialization, and assigns it an authorization callback of NONE. This variable contains the date and time at which the server was started on the node.

create proc *proc_name* **{***parameters***}** *auth_callback* **{***procedure***}**

The **create proc** statement creates a new procedure in the Sysctl server. This new procedure can be invoked from a client by supplying its name along with any defined parameters. For example:

```
create proc mydate {} AUTH {exec /bin/date}
```

creates the procedure **mydate** which has no parameters. This procedure has an authorization callback of AUTH. The procedure is comprised of a single statement (**exec** /**bin**/**date**).

create class *class_name class_file_name* **[***auth_callback***]**

The **create class** statement creates a class of commands in the Sysctl server. An optional authorization callback can be supplied. The authorization callback assigned to each object in the class is the logical OR of the class' callback (if supplied) and the object's callback. Thus, access to a class' object is granted if either the object's or the class' authorization callback allows access. For example:

```
create class sys $buildTop/samples/sysctl/sys.cmds
```

creates the class **sys**. If **$buildTop** is defined as /**usr**/**lpp**/**ssp**, the file /**usr**/**lpp**/**ssp**/**samples**/**sysctl**/**sys.cmds** contains the definition of the **sys** class.

include
The **include** statement includes the contents of another file in the configuration file. This provides a way of organizing the Server configuration statements into manageable groupings. For example:

```
include $buildTop/samples/sysctl/pdfpfck.cmds
```

causes the Sysctl server to read the contents of the specified file at initialization time.

set The **set** statement sets the value of the server variables ACL, LOG, or KEY, or sets values in the **env**() array. The default values for the Sysctl server's ACL file, log file, and service key file can be overridden by assigning values to the ACL, LOG, and KEY variables in the configuration file. For example, the following line overrides the default value for the log file name:

```
set LOG /var/sysctld.log_file
```

The values assigned to the ACL, LOG, and KEY variables are overridden by the optional command line arguments –**a**, –**l**, and –**k**.

Environment variables (such as the default PATH) can also be set within the configuration file by assigning values to the **env**() array. For example:

```
set env(PATH) /usr/bin:/bin:/usr/etc:/etc
```

sets the PATH for the Sysctl server. The **env**() array is assigned an authorization callback of SYSTEM which prevents its modification from outside the Sysctl server by a request sent by a Sysctl client.

load *lib_path* **[init_proc]**
The **load** command dynamically loads the shared library at *lib_path* into memory. If the **init_proc** parameter is given, it is used as the library's initialization procedure. Otherwise, the name of the initialization function is derived from the library name as follows:

```
library name    init function
------------    -------------
libxxx.sl       xxx_Init()
libxxx.a        xxx_Init()
```

The Sysctl server exports an API which the library uses to define commands, variables, authorization callbacks, and interpreter deletion callbacks. See the **load**() help page for details.

Other Sysctl Commands
The configuration file can also contain other Sysctl commands to modify the behavior of the server. For example, the following command in the configuration file causes the authorization callback for the **svcconnect** command to be changed from the default value of AUTH to NONE. This would allow nonauthenticated clients to connect to the server.

```
setauth -cmd svcconnect NONE
```

Prerequisite Information

See the description of Sysctl in *IBM Parallel System Support Programs for AIX: Administration Guide*.

Related Information

SP Command: **sysctl**

SP Daemon: **sysctld**

Chapter 3. DTMS Enablement Function I/O Control Requests

Note

This information on the distributed tape management system enablement facility is provided for backward compatibility purposes only.

We recommend the use of IBM Network Tape Access and Control System for AIX (NetTAPE). NetTAPE is a program that provides consolidated tape operations and transparent access to a wide variety of remote tape devices on an SP system.

This section describes the ioctls that are supported by the distributed tape management system (DTMS) enablement facility. Refer to the DTMS chapter in *IBM Parallel System Support Programs for AIX: Administration Guide* for additional information.

To access the DTMS online manual pages in this chapter and those in Chapter 3 (**mt_tmcp**, **mte_exits**, **mte_itable**, **mte_make**, and **mte_tmpcp**), set the MANPATH environment variable as follows:

for ksh

```
export MANPATH=$MANPATH:/usr/lpp/mte/man
```

for csh

```
setenv MANPATH $MANPATH\:/usr/lpp/mte/man
```

© Copyright IBM Corp. 1996

mt_cassign

Purpose

mt_cassign – Controls the assignment of a magnetic tape device.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSCA,***arg***);**
int *fildes***;**
struct tmsasgn ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSCA	Specifies the **ioctl()** request to assign (or unassign) a device.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call controls the assignment of a device attached through a control unit that supports multipath mode to this host system. When a device is successfully assigned to a host system, the password that was passed with this request is assigned to the device. Any host that provides the password used to assign a device may override the assignment and have the device reassigned to the overriding host. This **ioctl()** system call is also used to unassign a device.

A user or process must have the appropriate security privilege to use this function. This **ioctl()** cannot be issued against a tape management control point (TMCP) or a tape management pseudo control point (TMPCP). It can only be issued on a standard device or a device which was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode.

The assign/unassign protocol is supported only on 3480 and 3490 tape devices.

The **tmsasgn** structure is defined as follows:

```
struct tmsasgn {
int   resvd;                  /* Reserved                          */
int   versn;                  /* Version number field              */
int   secukey;                /* Security key                      */
char  subcmd;                 /* Subcommand field                  */
char  asg_status;             /* Field status                      */
char  asgn_pw[11];            /* Assignment password               */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_CASGVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support “tmsmpi” C2 Security privilege, this field is not used. If a security key is established on a system that does not support the “tmsmpi” C2 Security privilege, this field must contain the security key established by the MTTMSCSK **ioctl()**.
subcmd	**FORCE_ASSIGN**	Overrides the current assignment of this device (if currently assigned) and assigns this device to this host system. **Note:** Any operation in progress with the currently-attached host terminates with unpredictable results. This command should only be used in a controlled environment.
	PASSIVE_ASSIGN	Assigns this device to this host and assigns the password passed with this request if the device is not already assigned to another host system.
	UNASSIGN	Unassigns this device from this host.
asgn_pw		Assigns the password.

Return Values

On successful completion, the following values are returned in the **tmsasgn** structure:

resvd	**0**	Reserved.
versn	**TMS_CASGVSN**	Specifies the block structure version number passed with the request.
asg_status		Indicates the status of the operation.
	ASSIGN_ELSEWHERE	Indicates whether the device is already assigned to another host system. The operation was not successful (PASSIVE_ASSIGN).
	ASSIGNED	Indicates whether the device was successfully assigned to this host system.
	NOT_ASSIGNED	Indicates whether this request could not complete for reasons other than normal assignment to another host system (ASSIGN_ELSEWHERE). If the subcommand was FORCE_ASSIGN, the password may be invalid.
	UNASSIGNED	Indicates whether the device was successfully unassigned from this host system.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that the command was issued to a tape management control point (TMCP or TMPCP), a library management control point (LMCP), or a tape device opened in User Protocol Interface (UPI) mode.
[EINVAL]	Indicates that the security key is not valid.
[EIO]	Indicates that the I/O failed.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_cskey

Purpose

mt_cskey – Establishes a system security key for tape management **ioctl()** access. This security key is required on systems that do not support the “tmsmpi” C2 security privilege.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys**/**mttms.h**>

int ioctl(*fildes***,MTTMSCSK,***arg***);**
int *fildes***, MTTMSCSK;**
struct cskey ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSCSK	Specifies the **ioctl()** request to create a system security key.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call establishes a system security key required for Management Protocol Interface (MPI) **ioctl()** access on systems that do not support the “tmsmpi” C2 security privilege. It can only be issued against a tape management control point (TMCP) and must be issued immediately following the opening of the TMCP.

The **cskey** structure is defined as follows:

```
struct cskey {
int  resvd;         /* Reserved                                  */
int  versn;         /* Version number field                      */
int  cskstatus;     /* Security key status                       */
int  secukey;       /* Security key                              */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_SKEYVSN**	Specifies the block structure version number.
secukey		Specifies the system security key to be established and subsequently required for other Management Protocol Interface (MPI) **ioctl()** access.

Return Values

On successful completion, the following values are returned in the **cskey** structure:

resvd	**0**	Reserved.
versn	**TMS_SKEYVSN**	Specifies the block structure version number passed with the request.
cskstatus		Indicates the security key status.
	SK_C2	Indicates that the operating system supports the "tmsmpi" C2 Security privilege, therefore, the security key passed in the *secukey* field was not accepted.
	SK_EXISTS	Indicates that a security key was already created for this device driver. The security key passed in the *secukey* field was not accepted.
	SK_VALID	Indicates that the security key passed in the *secukey* field was established as a valid security key.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that either an invalid field was detected on the request, or that the *major/minor number* for this request did not have the MPI bit set.
[EINVAL]	Indicates that the security key is not valid.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.
[EPERM]	Indicates that the operating system supports the "tmsmpi" C2 privilege and that the requesting process does not have the "tmsmpi" privilege.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_cupi

Purpose

mt_cupi – Creates a user access path to a logical device image.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>
int ioctl(*fildes***,MTTMSCUPI,***arg***);**
int *fildes***, MTTMSCUPI;**
struct cupi ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSCUPI	Specifies the **ioctl()** request to create a user access path logical device image within a device driver.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call creates a logical device image. The logical device created represents an image of the physical device to which user requests can be issued. This image is referred to as the User Protocol Interface (UPI). Requests made to the UPI image must have the UPI bit set in the device minor number.

This **ioctl()** system call returns the content of the device major/minor numbers (with the UPI bit set) that can be used to create device special files and subsequently used to access the logical device image in UPI mode. It can only be issued against a device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device, or against a tape management pseudo control point (TMPCP) device (please refer to the **mte_cupi ioctl()** system call for more information on TMPCP logical device creation).

The **cupi** structure is defined as follows:

```
struct cupi {
int    resvd;      /* Reserved                                    */
int    versn;      /* Version number field                        */
int    secukey;    /* Security key                                */
int    cupiopt;    /* Options field, for future use               */
dev_t  devmmn;     /* Device major/minor number                   */
dev_t  cupimm1;    /* Major/minor number assigned to this entry   */
dev_t  cupimm2;    /* Major/minor number assigned to this entry   */
dev_t  cupimm3;    /* Major/minor number assigned to this entry   */
char   cupitype;   /* Logical device image type                   */
int    cupideva;   /* Physical device address                     */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_CUPIVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the "tmsmpi" C2 Security privilege, this field is not used. On systems that do not support the "tmsmpi" C2 Security privilege, this field must contain the security key established by the MTTMSCSK **ioctl()**.
cupiopt		Reserved.
devmmn	**0**	Indicates to use the *major/minor number* of the device that this **ioctl()** is issued against.
	major/minor number	Specifies the *major/minor number* for the device that this UPI image is being created for. This entry is required if the request is issued against a tape management control (TMCP) point. Usage note: *1* For requests issued against a TMCP, the *major/minor number* value contained in this field is normally obtained from the *mjrmin1* field of the device configuration entries returned from the MTTMSGDCI **ioctl()** command issued to the same TMCP.
cupitype		Specifies the logical device image type to be created.
	MAGTAPE	Specifies the logical device image for a magnetic tape device. The *major/minor number* of the device is passed in the *devmmn* field. This logical device image type is supported by the tape device driver. Three UPI device *major/minor numbers* are returned for this logical device image.
cupideva	*physical device address*	Specifies the *physical device address* represented by the logical device image.

Return Values

On successful completion, a User Protocol Interface logical device image is created. The fields contain the following values when this system call completes successfully.

On return, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_CUPIVSN**	Specifies the block structure version number passed with the request.
cupimm1	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* is *no rewind* and *no unload*.
cupimm2	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* is *rewind*.
cupimm3	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* is *rewind* and *unload*.

A return value of 0 indicates a successful operation. A return value of -1 indicates the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EAGAIN] Indicates that a socket interface was not created due to resource limitations.

[EBUSY] Indicates that someone else is using the device specified in *fildes*.

[EINVAL] Indicates that the version field is out of date. Recompile the application.

[EINVAL] Indicates that the command was issued to a standard device or to a UPI device.

[EINVAL] Indicates that the *devmmn* field points to the TMCP.

[EINVAL] Indicates that the command was issued to a TMCP and *devmmn* is 0.

[EINVAL] Indicates that the *cupitype* is invalid for the device.

[EINVAL] Indicates that the security key is not valid.

[ENOMEM] Indicates that no memory is available for the UPI control block.

[ENOTTY] Indicates that a logical device image already exists for this device. The request is not executed.

[ENXIO] Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_gdci

Purpose

mt_gdci – Returns tape device configuration information.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>
int ioctl(*fildes***,MTTMSGDCI,***arg***);**
int *fildes***, MTTMSGDCI;**
struct devconfh ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSGDCI	Specifies the **ioctl()** request to return device configuration information.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call returns configuration information for devices configured to the tape device driver. The number of configuration entries returned is based on the subcommand issued, the configuration entry size, the size of the buffer passed with the request, and the number of entries that remain to be presented.

The major/minor numbers for the device, the configuration disposition, the stanza name, and the device class are returned for all devices. For LMCP logical devices (the device class is LMCP_cl), additional fields returned are the *lmcp_type*, the library sequence number, and the major/minor numbers that can be used to create the special file needed to access the logical device. For tape devices (the device class is MAGTAPE), all **devconf** structure fields are returned, except for the *lmcp_type*; the second set of major/minor numbers are the values that can be used to create the special file needed to access the device in MPI mode (*mjrmin2*).

This **ioctl()** system call can only be issued against a device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode.

The **devconfh** and **devconf** structures are defined in **mttms.h** as follows:

```
struct devconfh {
int    resvd;              /* Reserved                                */
int    versn;              /* Version number field                    */
int    secukey;            /* Security key                            */
char   subcmd;             /* Subcommand field                        */
int    bfrln;              /* Length in bytes of data return area     */
int    resplngth;          /* Length in bytes of the data returned    */
struct devconf *devconf_p; /* Pointer to data return area             */
};
struct devconf {
int    resvd;              /* Reserved                                */
int    versn;              /* Version number field                    */
dev_t  mjrmin1;            /* Major/minor device number               */
dev_t  mjrmin2;            /* MPI major/minor device number           */
int    devaddr;            /* Physical device address                 */
char   devfeat;            /* Device features, config. specified      */
char   autofeat;           /* Device features, auto. defined          */
ushort devfmts;            /* Device recording formats                */
ushort devtypn;            /* Device type number                      */
char   devmodn;            /* Device model number                     */
char   devatt;             /* Device attachment type                  */
char   config_disp;        /* Configuration disposition               */
char   devclass;           /* Class of device                         */
char   mediatype;          /* Type of media supported by device       */
char   lmcp_type;          /* Type of LMCP (devclass is LMCP_cl)      */
char   lib_subid;          /* Library subsystem ID                    */
char   lib_seqno;          /* Library sequence number                 */
ulong  host_ipa;           /* Host IP address                         */
char   related_dev[32];    /* Related device name                     */
char   stanzan[32];        /* Stanza name                             */
};
```

On request, the **devconfh** structure contains:

resvd	**0**	Reserved.
versn	**TMS_GDCIVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support “tmsmpi” C2 Security privilege, this field must contain the security key established by the MTTMSCSK **ioctl()**.
subcmd	**GETFIRST**	Returns device configuration information for this device driver starting with the first configuration entry.
	GETNEW	Returns device configuration information for this device driver starting with the first added, closed, or changed entry that was not previously returned since a change in state had occurred for the configuration entry. If no new device configuration was made, the *resplngth* field is 0 and the return value is 0.
	GETNEXT	Returns device configuration information for this device driver starting with the next entry following the last entry returned in a prior GETFIRST or GETNEXT request.
brfln	*data size*	Specifies the size in bytes of the data return area.
devconf_p	*pointer*	Specifies the pointer to the data return area.

Return Values

The fields contain the following values when this system call completes successfully:

resvd	**0**	Reserved.
versn	**TMS_GDCIVSN**	Specifies the block structure version number.
resplngth	*length*	Specifies the size in bytes of the total configuration information returned.
devconf_p	*pointer*	Specifies the pointer to the data return area.

Each configuration information entry structure (**devconf**) contains the following:

resvd	**0**	Reserved.
versn	**TMS_GDCIVSN**	Specifies the block structure version number.
mjrmin1	*major/minor number*	Specifies the *major/minor number* assigned to this device.
mjrmin2	*major/minor number*	Indicates that when the device class is MAGTAPE, this field contains the *major/minor number* required to access this device in MPI mode. When device class is LMCP_cl, this field contains the *major/minor number* required to access this logical device in LPI mode.
devaddr		Specifies the physical device address.
devfeat		Specifies the device features specified by the **odm** attribute (if any) for this device. The defined entries include:
	TMS_ALT_PATH	Indicates that this device is sharable using the assign/unassign protocol. This device can be accessed by this tape management system through more than one host node/access path. This access path should be used as an alternate access path for this device by this tape management system.
	TMS_MTFEAT1	Reserved.
	TMS_MTFEAT2	Reserved.
	TMS_MTFEAT3	Reserved.
	TMS_MTFEAT4	Reserved.
	TMS_MTFEAT5	Reserved.
	TMS_MTSHARE	Indicates that this device is sharable (with other host tape management systems) using the assign/unassign protocol. This is the only known path to this device to this tape management system. The device can also be known to other (atomic) tape management systems.
	TMS_PRI_PATH	Indicates that this device is sharable using the assign/unassign protocol. This device can be accessed by this tape management system through more than one host node/access path. This access path should be used as the primary access path for this device by this tape management system.
autofeat		Indicates the device features determined by the device.
	TMS_MTACL	Indicates that the device has the automatic cartridge loader feature.
devfmts		Indicates the recording formats supported by the device.
	MTGCR	Indicates that the device supports the GCR recording format.
	MTNRZI	Indicates that the device supports the NRZI recording format.
	MTNT1_BAS	Indicates that the device supports the NT1 basic recording format.

	MTNT1_COM	Indicates that the device supports the NT1 compacted recording format.
	MTNT2_BAS	Indicates that the device supports the NT2 basic recording format.
	MTNT2_COM	Indicates that the device supports the NT2 compacted recording format.
	MTPE	Indicates that the device supports the PE recording format.
	MT3480_BAS	Indicates that the device supports the 3480 basic recording format.
	MT3480_COMP	Indicates that the device supports the 3480 compacted recording format.
	MT3490_BAS	Indicates that the device supports the 3490_BAS recording format.
	MT3490_COMP	Indicates that the device supports the 3490 compacted recording format.
	MT4MM_BAS	Indicates that the device supports the 4-mm basic recording format.
	MT4MM_COM	Indicates that the device supports the 4-mm compacted recording format.
	MT8MM_BAS	Indicates that the device supports the 8-mm basic recording format.
	MT8MM_COM	Indicates that the device supports the 8-mm compacted recording format.
devtypn		Specifies the device type number. Device types include:
	IBM3420	Specifies a 3420 device type.
	IBM3422	Specifies a 3422 device type.
	IBM3430	Specifies a 3430 device type.
	IBM3480	Specifies a 3480 device type.
	IBM3490	Specifies a 3490 device type.
	MTNT1	Specifies a IBMNT1 device type.
	MTNT2	Specifies a IBMNT2 device type.
	MT4MM	Specifies a 4-mm device type.
	MT8MM	Specifies an 8-mm device type.
devmodn	*nnnn*	Specifies the model number of the device listed in the *devtypn* field.
devatt		Specifies the device attachment type used to attach this device to this host.
	R6KE1	Indicates that this device is attached through an RS/6000 390 ESCON* channel attachment card.
	R6KP1	Indicates that this device is attached through an RS/6000 390 Parallel channel attachment card.
	R6KS1	Indicates that this device is attached through an RS/6000 SCSI-1 channel attachment card.
	R6KS2	Indicates that this device is attached through an RS/6000 SCSI-2 channel attachment card.
	S390	Indicates that this device is attached through a System/390 subchannel (parallel or serial channel attachment).
config_disp		Indicates the disposition of the configuration information that is returned.
	CNGDISP	Indicates that this device existed in the configuration, but a feature was changed.

	MPIC	Indicates that this device is in the configuration and is in a forced close (MPI) condition.
	NEWDISP	Indicates that this is a new device added to the device driver configuration.
	REMDISP	Indicates that there was a request to remove this device from the device driver configuration.
	SAMEDISP	Indicates that this device was in the configuration and there is no change associated with this device.
	UNCNFGD	Indicates that this device was unconfigured.
devclass		Indicates a device class entry.
	MAGTAPE	Indicates a magnetic tape device.
	LMCP_cl	Indicates a library management control point logical device.
mediatype		Specifies the supported media type. The media type can be either defined by the device type (for example, 3420 = TMS_REEL), or it can be explicitly defined by the TMS_Media_Type attribute in the **odm** configuration stanza entry for this device.
	TMS_CART	Indicates a cartridge type device (3480, 3490).
	TMS_DLCART	Indicates a double length cartridge type device (3490E).
	TMS_MEDIA1	Reserved.
	TMS_MEDIA2	Reserved.
	TMS_MEDIA3	Reserved.
	TMS_MEDIA4	Reserved.
	TMS_REEL	Indicates a reel type device (3420).
	TMS_SMCART	Indicates a small cartridge type device.
	TMSNT1	Indicates a NT1 type device.
	TMSNT2	Indicates a NT2 type device.
	TMS4MM	Indicates a 4-mm type device.
	TMS8MM	Indicates an 8-mm type device.
lmcp_type	*type*	Indicates the type of LMCP port provided by this logical device. Possible values are: • MT_LMCP_CMD_PORT • MT_LMCP_NTF_PORT0 • MT_LMCP_NTF_PORT1 • MT_LMCP_NTF_PORT2 • MT_LMCP_NTF_PORT3 • MT_LMCP_NTF_PORT4 • MT_LMCP_NTF_PORT5 • MT_LMCP_NTF_PORT6 • MT_LMCP_NTF_PORT7 • MT_LMCP_NTF_PORT8 • MT_LMCP_NTF_PORT9 • MT_LMCP_NTF_PORT10 • MT_LMCP_NTF_PORT11 • MT_LMCP_NTF_PORT12 • MT_LMCP_NTF_PORT13 • MT_LMCP_NTF_PORT14
lib_subid	*id*	Specifies the subsystem ID provided by a tape library device.

lib_seqno	*number*	Specifies the library sequence number uniquely identifying a tape library and each of its associated tape devices and LMCP logical devices.
host_ipa	*number*	Specifies the IP address (host ID in hexadecimal) of the host to which this device is attached.
related_dev	*name*	Specifies the global name reference for this device. This is the name provided in the stanza entry used to relate an alternate path device attached to multiple host nodes. If the global name contained in the **odm** attribute is longer than 32 bytes, it is truncated to 32 bytes.
stanzan	*name*	Specifies the stanza name for the device. If the stanza name is longer than 32 bytes, it is truncated to 32 bytes.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL] Indicates that the version field is out of date. Recompile the application.

[EINVAL] Indicates that the command was issued to a standard device or UPI device.

[EINVAL] Indicates that the security key is not valid.

[ENOSPC] Indicates that there is insufficient room in the data return area to provide one configuration information entry. Expand the data return area and retry the operation.

[ENXIO] Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_rdbd

Purpose

mt_rdbd – Reads 3480 or 3490 buffer data.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***, MTTMSRDBD,** *arg***);**
int *fildes***, MTTMSRDBD;**
struct tmsrdbd ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSRDBD	Specifies the **ioctl()** request to read buffered data from a 3480 or 3490 tape control unit.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call reads data contained in the data buffer for a 3480 or 3490 tape device. It can only be issued against a tape device operating in Management Protocol Interface (MPI) mode.

This **ioctl()** request cannot be issued against a tape management control point (TMCP) or a tape management pseudo control point (TMPCP).

The **tmsrdbd** structure is defined as follows:

```
struct tmsrdbd {
int    resvd;            /* Reserved                              */
int    versn;            /* Version number field                  */
int    secukey;          /* Security key                          */
addr   rdbd_addr;        /* Data address                          */
ulong  rdbd_cnt_to_rd;   /* Data count                            */
ulong  rdbd_cnt_read;    /* Bytes read                            */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_RDBDVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. If a security key is established on a system that does not support the “tmsmpi” C2 Security privilege, this field must contain the security key established by the MTTMSCSK **ioctl()**.

rdbd_addr	*data address*	Contains the user data address that the buffered data is to be read into.
rdbd_cnt_to_rd	*data count*	Contains the byte count of the data to be read.

Return Values

On successful completion, the following values are returned in the **tmsrdbd** structure:

resvd	**0**	Reserved.
versn	**TMS_RDBDVSN**	Specifies the block structure version number passed with the request.
rdbd_cnt_read	*byte count read*	Contains the number of bytes of data read.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that the version field is not up to date. Recompile the application.
[EINVAL]	Indicates that the command was issued to a TMCP or TMPCP.
[EINVAL]	Indicates that the command was issued to a device that does not support the read data buffer operation.
[EINVAL]	Indicates that the security key is not valid.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.
[EPERM]	Indicates that the user does not have the “tmsmpi” privilege on a system that supports C2 Security.
[EPERM]	Indicates that the user did not supply the correct security key on a system that does not support C2 Security.
[EPERM]	Indicates that the user does not have the MPI bit set in the device minor number.
[EPERM]	Indicates that the device was not opened in Management Protocol Interface (MPI) mode.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_rupi

Purpose

mt_rupi – Removes a user access path to a logical device image from the device driver configuration.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSRUPI,***arg***);**
int *fildes***, MTTMSRUPI;**
struct rupi ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSRUPI	Specifies the **ioctl()** request to remove a user access path logical device entry from the device driver configuration.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call removes a logical device image created by a previous MTTMSCUPI **ioctl()** system call. It can only be issued against a device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device or against a tape management pseudo control point (TMPCP) device (please refer to the **mte_rupi ioctl()** system call for more information on TMPCP logical device removal).

The **rupi** structure is defined as follows:

```
struct rupi {
int     resvd;     /* Reserved                                  */
int     versn;     /* Version number field                      */
int     secukey;   /* Security key                              */
dev_t   upimm;     /* Device major/minor number                 */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_VERSN**	Specifies the block structure version number.

secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established by the MTTMSCSK **ioctl()**.
upimm	**0**	Indicates to use the *major/minor number* of the UPI device that this **ioctl()** is issued against.
	major/minor number	Specifies the MPI *major/minor number* for the device this UPI is being removed from. This entry is required if the request is issued against a tape management control point (TMCP).

Return Values

On successful completion, the UPI logical device image is removed.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that either no device was specified for the TMCP request, or that a wrong version number was used.
[EINVAL]	Indicates that the security key is not valid.
[ENOTTY]	Indicates that either the UPI does not exist, is in use, or is in the wrong state.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_srlf

Purpose

mt_srlf – Sets or reads last failure information and tape management status for a magnetic tape device.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSSRLF,***arg***)**
int *fildes***, MTTMSSRLF;**
struct tmsmi ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSSRLF	Specifies the **ioctl()** request to set or read magnetic tape device failure information.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call sets or reads failure information and potentially tape management status (if applicable) associated with a logical device image. It can be issued against a tape device not operating under the domain of a tape management system (standard device access), or a tape device operating in Management Protocol Interface (MPI) mode or User Protocol Interface (UPI) mode.

A user or process must have the appropriate security privilege to use the “set” subcommands of this function. The “set” function is used to control the last failure indicator and tape management status field that can be returned on a User Protocol Interface (UPI) device. This **ioctl()** system call cannot be issued against a tape management control point (TMCP) or a tape management pseudo control point (TMPCP).

The **tmsrlf** structure is defined as follows:

```
struct tmsrlf {
int     resvd;              /* Reserved                              */
int     versn;              /* Version number field                  */
int     secukey;            /* Security key                          */
char    subcmd;             /* Subcommand field                      */
char    lf_status;          /* Field status                          */
ushort  last_dev_fail;      /* Last device failure indication        */
ushort  tape_mgt_status;    /* Tape management system status         */
};
```

On request, the fields contain:

resvd **0** Reserved.

versn	**TMS_SRLFVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSK **ioctl()**, if the subcommand is SET_LDF, SET_TMS, or SET_ALL.
subcmd	**RD_ALL**	Returns the last failure indicator and tape management status field for this device.
	SET_ALL	Sets both the *last_dev_fail* field and turns on the LDF_VALID indicator in the *lf_status* field, and sets the *tape_mgt_status* field and turns on the TMS_VALID indicator in the *lf_status* field. This subcommand can only be issued by a user or process that opened the device in MPI mode.
	SET_LDF	Sets the *last_dev_fail* field and turns on the LDF_VALID indicator in the *lf_status* field. This subcommand allows a tape management system to control the failure indication that can be returned on the User Protocol Interface (UPI) for this logical device. This subcommand can only be issued by a user or process that opened the device in MPI mode.
	SET_TMS	Sets the *tape_mgt_status* field and turns on the TMS_VALID indicator in the *lf_status* field. This subcommand allows a tape management system to control the tape management status that can be returned on the User Protocol Interface (UPI) for this logical device. This subcommand can only be issued by a user or process that opened the device in MPI mode.

The following fields are used on input when the subcommand field is SET_LDF, SET_TMS, or SET_ALL:

last_dev_fail	Specifies the last device failure indicator. When the subcommand field is SET_LDF or SET_ALL, the value supplied in this field is returned on a subsequent SRLF **ioctl()** command issued against this User Protocol Interface (UPI) device, unless a new device failure is detected before the request is received.
tape_mgt_status	Specifies the tape management status indicator. When the subcommand field is SET_TMS or SET_ALL, the value supplied in this field is returned on a subsequent SRLF **ioctl()** command issued against this User Protocol Interface (UPI) device.

Return Values

On successful completion, when subcommand contains RD_LF, the following values are returned in the **tmsrlf** structure:

resvd	**0**	Reserved.
versn	**TMS_SRLFVSN**	Specifies the block structure version number passed with the request.
subcmd	**RD_ALL**	Returns the last failure indicator and tape management status for this device.
lf_status		Contains the field validity status, such as, the current status of the *last_dev_fail* and *tape_mgt_status* fields. The individual status bits are defined as follows:
	LDF_VALID	Indicates that the *last_dev_fail* field contains unreported status. This indicator is set to 1 when new failure status is received for this device. This indicator is reset to 0 on presentation of status and remains reset until new failure status is received.

TMS_VALID Indicates that the *tape_mgt_status* field contains unreported status. This indicator is set to 1 when new tape management status is received for this device. This indicator is reset to 0 on presentation of status and remains reset until new tape management status is received. This status indicator is only returned on requests directed against a User Protocol Interface (UPI) device.

UPI_ACCESS Indicates that an SRLF **ioctl()** was received for this User Protocol Interface (UPI) device since the last time a SET_LDF or SET_TMS subcommand was issued for this device. This status indicator is only returned on requests directed against a Management Protocol Interface (MPI) device.

last_dev_fail Contains the last device failure indicator. The value returned in this field represents the last failure condition detected by the tape device driver for this device. The actual value returned is device-type dependent and, therefore, can change as support for additional devices is added.

For IBM 3480 and 3490 type devices, the indicator is in the form of the "Error Recovery Procedure Action (ERPA) Code" as documented in sense bytes 3 and 4 for these devices. Refer to *IBM 3480 Magnetic Tape Subsystem Reference: Channel Commands, Status and Sense Bytes, and Error Recovery Procedures* and to *IBM 3490 Magnetic Tape Subsystem Reference: Channel Commands, Status and Sense Bytes, and Error Recovery Procedures* to obtain descriptions of the codes returned.

For IBM 3420 type devices, detected errors are mapped to the appropriate IBM 3480/3490 ERPA code.

tape_mgt_status Contains the tape management status indicator. For User Protocol Interface (UPI) devices, this field can be used to allow a tape management system to convey operation or error status relative to tape management system activity. The values returned in this field are defined by the tape management system.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following value:

[EINVAL] Indicates that the version field is not up to date. Recompile the application.

[EINVAL] Indicates that the subcommand field value is undefined.

[EINVAL] Indicates that the command was issued to a TMCP or TMPCP.

[EINVAL] Indicates that the security key is not valid.

[ENXIO] Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

[EPERM] Indicates that the subcommand is SET_LDF, SET_TMS, or SET_ALL and the user does not have the "tmsmpi" privilege on a system that supports C2 Security.

[EPERM] Indicates that the subcommand is SET_LDF, SET_TMS, or SET_ALL and the user did not supply the correct security key on a system that does not support C2 Security.

[EPERM] Indicates that the subcommand is SET_LDF, SET_TMS, or SET_ALL and the user does not have the MPI bit set in the device minor number.

[EPERM] Indicates that the subcommand is SET_LDF, SET_TMS, or SET_ALL and the device was not opened in Management Protocol Interface (MPI) mode.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_srmi

Purpose

mt_srmi – Sets, changes, or reads mode information of a logical magnetic tape device.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSSRMI,***arg***);**
int *fildes***, MTTMSSRMI;**
struct tmsmi ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSSRMI	Specifies the **ioctl()** request to set, change, or read logical magnetic tape device mode information.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call sets, changes, or reads mode information associated with a logical device image created by a previous **MTTMSCUPI ioctl()** system call. This logical device image is referred to as the User Protocol Interface (UPI) in the documentation. The various information fields associated with this request are used to control the activity of the UPI logical device image.

This **ioctl()** system call can only be issued against a device that was opened in Management Protocol Interface (MPI) mode. It cannot be issued against a tape management pseudo control point (TMPCP).

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device.

The **tmsmi** structure is defined as follows:

```
struct tmsmi {
int    resvd;         /* Reserved                                    */
int    versn;         /* Version number field                        */
int    secukey;       /* Security key                                */
char   subcmd;        /* Subcommand field                            */
char   lpm;           /* Label processing mode                       */
ushort setdisp;       /* Disposition indicator                       */
ushort usrdisp;       /* User selected disposition at open           */
dev_t  upimm;         /* Major/minor number of the logical device    */
int    blksize;       /* Record/block size parameter                 */
int    recmode;       /* Recording mode format                       */
char   lblform;       /* Label format type                           */
char   accmode;       /* Media access mode                           */
char   rwmode;        /* Read/write access mode                      */
char   openrtn;       /* Open return notification mode               */
char   wrtrtn;        /* Write return notification mode              */
char   rdrtn;         /* Read return notification mode               */
int    mtoflags;      /* Open() O_Flags flags                        */
int    mtbidstr;      /* Starting block ID number                    */
int    mtbidend;      /* Ending block ID number                      */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_SRMIVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSK **ioctl()**.
subcmd	**NOTAVAIL**	Indicates that this device is not available for use through the User Protocol Interface (UPI). Any UPI requests received for this logical device causes the tape manager to be notified with the UPINA (*ddreason*) reason code. Refer to the **mt_srsi** command for definitions of the *ddreason* codes.
	RDMODE	Returns the current mode information for this User Protocol Interface device.
	STMODE	Sets the label processing modes, format types selection, and the disposition mode selection for this User Protocol Interface device. It also marks this UPI logical device as available for use.
upimm		Specifies the *major/minor number* of the UPI device.
	0	Specifies the UPI address associated with the physical device that corresponds with the *major/minor number* of the device.
	major/minor number	Specifies the *major/minor number* of the device.

STMODE FIELDS:

The following fields are only used when the *subcmd* field is STMODE:

lpm	**BLP_LPM**	Indicates bypass label processing mode.
	NL_LPM	Indicates no label processing mode.
	SLA_LPM	Indicates ANSI standard label processing mode.
	SLI_LPM	Indicates IBM standard label processing mode.

setdisp	**NRAC**	Indicates not to rewind or unload the tape volume at **close()**.
	RAC	Rewinds the tape volume at **close()**.
	UAC	Unloads the tape volume at **close()**.
	USERDISP	Allows the user to select disposition at open for this UPI device.
lblform	**EQUAL_LBL**	Indicates a label format other than type *U*. The user of this UPI device can only write records of the length set in the *blksize* field. An attempt to write any other record length causes the Set/Read State Information (SRSI) **ioctl()** to present WRTNBS as the *ddreason* code.
	MAX_LBL	Indicates a label format other than type *U*. The user of this UPI device can only write records of a length equal to or less than that set in the *blksize* field. An attempt to write a record length greater than the *blksize* field causes the Set/Read State Information (SRSI) **ioctl()** to present WRTNBS as the *ddreason* code.
	MULTI_LBL	Indicates a label format other than type *U*. The user of this UPI device can only write records of the length equal to or a multiple of that set in the *blksize* field. An attempt to write a record length not equal to or a multiple of the *blksize* field causes the Set/Read State Information (SRSI) **ioctl()** to present WRTNBS as the *ddreason* code.
	U_LBL	Indicates a *U* label format type. When this format is specified, the user can read or write any length records.
blksize	**blksize**	Indicates the actual or maximum record size that can be written to or requested from tape. The use is determined by the *lblform* field.
accmode		Indicates media access mode.
	D_ACC	Indicates data set access mode.
	V_ACC	Indicates volume access mode.
rwmode	**RDONLY_MODE**	Indicates read-only mode. Any **open()** that specifies write only or read/write is rejected with **errno** set to [EINVAL].
	ROWNE_MODE	Indicates read-only mode. The volume write capability should not be enabled. Any **open()** that specifies write only or read/write is rejected with **errno** set to [EINVAL]. If the volume has the write capability enabled, the tape manager is notified with the write enabled volume (WEV) *ddreason* code. Refer to the **mt_srsi** command for definitions of the *ddreason* codes.
	RW_MODE	Indicates read/write mode. Any **open()** that specifies read or write is accepted.
	WRTONLY_MODE	Indicates write-only mode. Any **open()** that specifies read-only or read/write is rejected with **errno** set to [EINVAL].
openrtn	**MT_ALWAYS**	Indicates that the tape manager is notified with the UPIO *ddreason* code when any **open()** is received for this device.
	MT_NOTALWAYS	Indicates that the tape manager is not notified when any **open()** is received for this device.
wrtrtn	**MT_ALWAYS**	Indicates that the tape manager is notified with the user write (WRTNTFY) *ddreason* code when any **write()** is received for this device.
	MT_NOTALWAYS	Indicates that the tape manager is not notified when any **write()** is received for this device.
rdrtn	**MT_ALWAYS**	Indicates that the tape manager is notified with the user read (RDNTFY) *ddreason* code when any **read()** is received for this device.
	MT_NOTALWAYS	Indicates that the tape manager is not notified when any **read()** is received for this device.

mtbidstr	*nnnn*	Indicates the lowest starting block ID number that is accepted in a locate block **ioctl()** that is received for this UPI device.
	X'FFFFFFFF'	Indicates that a locate block **ioctl()** is not be accepted for this UPI device and the requestor receives an **errno** value of [EINVAL].
mtbidend	*nnnn*	Indicates that the highest ending block ID number that is accepted in a locate block **ioctl()** that is received for this UPI device.
	X'00000000'	Indicates that a locate block **ioctl()** is not accepted for this UPI device and that the requestor receives an **errno** value of [EINVAL].

Return Values

On successful completion, when *subcmd* contains RDMODE, the following values are in the **tmsmi** structure:

resvd	**0**	Reserved.
versn	**TMS_SRMIVSN**	Specifies the block structure version number passed with the request.
subcmd	**RDMODE**	Returns the current statistical information for this device.
upimm		Specifies the *major/minor number* of the UPI device.
	0	Specifies the UPI address associated with the physical device that corresponds with the *major/minor number* of the device.
	major/minor number	Specifies the *major/minor number* of the device.
lpm		Specifies the label mode passed in the previous STMODE.
setdisp		Specifies the disposition passed in the previous STMODE.
usrdisp	**0**	Indicates if the special device file is not opened.
	NRAC	Indicates if opened as no rewind or unload at **close()**.
	RAC	Indicates if opened as rewind at **close()**.
	UAC	Indicates if opened as unload at **close()**.
recmode	**MTNT1_BAS**	Indicates that the device is operating in NT1 basic recording format.
	MTNT1_COM	Indicates that the device is operating in NT1 compacted recording format.
	MTNT2_BAS	Indicates that the device is operating in NT2 basic recording format.
	MTNT2_COM	Indicates that the device is operating in NT2 compacted recording format.
	MT4MM_BAS	Indicates that the device is operating in 4-mm basic recording format.
	MT4MM_COM	Indicates that the device is operating in 4-mm compacted recording format.
	MT8MM_BAS	Indicates that the device is operating in 8-mm basic recording format.
	MT8MM_COM	Indicates that the device is operating in 8-mm compacted recording format.
	RM_NOT_INIT	Indicates that the device recording format is uninitialized.
	RM_800	Indicates that the device is operating in 800 BPI recording format.
	RM_1600	Indicates that the device is operating in 1600 BPI recording format.
	RM_3480_BAS	Indicates that device is operating in 3480 standard recording format.
	RM_3480_COMP	Indicates that the device is operating in 3480 IDRC recording format.
	RM_3490_BAS	Indicates that the device is operating in 3490 standard recording format.
	RM_3490_COMP	Indicates that the device is operating in 3490 IDRC recording format.
	RM_6250	Indicates if the device is operating in 6250 BPI recording format.
lblform		Specifies the label format passed in the previous STMODE.

blksize		Specifies the block size passed in the previous STMODE.
accmode		Specifies the media access mode passed in the previous STMODE.
rwmode	**RDONLY_MODE**	Indicates read-only mode. Any **open()** that specifies write only or read/write is rejected with **errno** set to [EINVAL].
	ROWNE_MODE	Indicates read-only mode. The volume write capability should not be enabled. Any **open()** that specifies write only or read/write is rejected with **errno** set to [EINVAL]. If the volume has the write capability enabled, the tape manager is notified with the write enabled volume (WEV) *ddreason* code. Refer to the **mt_srsi** command for definitions of the *ddreason* codes.
	RW_MODE	Indicates read/write mode. Any **open()** that specifies read or write is accepted.
	WRTONLY_MODE	Indicates write-only mode. Any **open()** that specifies read only or read/write is rejected with **errno** set to [EINVAL].
openrtn		Indicates the **open()** return passed in the previous STMODE.
wrtrtn		Indicates the **write()** return passed in the previous STMODE.
rdrtn		Indicates the **read()** return passed in the previous STMODE.
mtoflags		Indicates the O_Flags passed by the last UPI **open()** system call. If the UPI device is not open or it is not currently processing an **open()** system call, this field contains -1.
mtbidstr		Specifies the starting block ID number passed in the previous STMODE command.
mtbidend		Specifies the ending block ID number passed in the previous STMODE command.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that the version field is not up to date. Recompile the application.
[EINVAL]	Indicates that the subcommand field value is undefined.
[EINVAL]	Indicates that the subcommand field contains STMODE and an invalid value was passed in the **tmsmi** structure.
[EINVAL**]**	Indicates that the command was issued to a TMCP or TMPCP and the *upimm* field is 0.
[EINVAL]	Indicates that the security key is not valid.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_srsi

Purpose

mt_srsi – Sets and reads state information of a logical magnetic tape device.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSSRSI,***arg***);**
int *fildes***,MTTMSSRSI;**
struct tmssi ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSSRSI	Specifies the **ioctl()** request to set or read state information of a logical magnetic tape device entry.
arg	Specifies the address of the **tmssi** structure passed.

Description

This **ioctl()** system call sets and reads state information associated with a logical device image created by a previous MTTMSCUPI **ioctl()** system call. The logical device image is referred to as the User Protocol Interface (UPI) in the documentation.

This **ioctl()** system call can only be issued against a device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device or against a tape management pseudo control point (TMPCP) device (refer to the **mte_srsi ioctl()** system call for more information on TMPCP state information access).

The **tmssi** structure is defined in **mttms.h** as follows:

```
struct tmssi {
int   resvd;      /* Reserved                                        */
int   versn;      /* Version number field                            */
int   secukey;    /* Security key                                    */
char  subcmd;     /* Subcommand field                                */
char  ddaction;   /* Device driver action code                       */
char  ddreason;   /* Device driver reason code                       */
char  pwait;      /* Process wait (suspended) indicator              */
char  pwaitp;     /* Process wait pending notification indicator     */
char  ddrcnt;     /* ddreason count                                  */
int   tminterrno; /* int/errno value to be returned to user          */
dev_t upimm;      /* Major/minor number of the logical device        */
int   ddinterrno; /* Device driver generated int/errno value         */
int   ioctlcnt;   /* ioctl() repetition count                        */
char  curscst;    /* Current system call state                       */
char  priscst;    /* Prior system call state                         */
char  volumest;   /* Volume state                                    */
char  lastmove;   /* Last volume movement direction                  */
int   volpostm;   /* Current volume position (# of tape marks)       */
int   volposrc;   /* Current vol. pos. (# of records from tape mark) */
int   pvpostm;    /* Prior volume position (# of tape marks)         */
int   pvposrc;    /* Prior volume pos. (# of records from tape mark) */
int   phvolpos;   /* Current physical volume position(3480/3490)     */
u_short timeOut;  /* Wait timeout value                              */
int   mtoflags;   /* O_Flags passed at open                          */
char  siflags;    /* Miscellaneous state indication field            */
char  purgeddr;   /* Purge ddreason field                            */
};
```

On request, the fields contain:

resvd **0** Reserved.

versn **TMS_SRSIVSN** Specifies the block structure version number.

secukey Specifies the system security key. On systems that support the "tmsmpi" C2 Security privilege, this field is not used. On systems that do not support the "tmsmpi" C2 Security privilege, this field must contain the security key established with the MTTMSCSK **ioctl()**.

subcmd

PACTIONR Performs the action indicated in the *ddaction* field and returns when complete.

PACTIONW Performs the action indicated in the *ddaction* field and waits. It only returns when an error or exception condition is detected.

PACTIONWT Performs the action indicated in the *ddaction* field and waits. It returns when an error or exception condition is detected or following a timeout period specified in the timeOut field. The default is 10 minutes.

RDSENSE Performs device sense operations and obtains physical block position information, if applicable. This request returns updated state information for this User Protocol Interface device based on the actual (current) state of the volume or device.

Note: This operation should only be requested when a UPI process is not active or when a UPI process is suspended (waiting).

RDSTATE Returns the updated state information for this User Protocol Interface device.

SRSIWAIT Waits. It only returns when an error or exception condition is detected.

	WAITTIME	Waits. It returns when an error or exception condition is detected or following a timeout period specified in the timeOut field. The default is 10 minutes.
ddaction		Contains the action code to be performed by the device driver if the *subcmd* is PACTIONR, PACTIONWT, or PACTIONW.
	MT_CONTINUE	Continues the last UPI user operation. **Note:** If the last UPI user operation was an **ioctl()**, the repetition count, if any, is that value previously returned or updated in the *ioctlcnt* field.
	MT_RETRY	Retries the last UPI user operation. **Note:** If the last UPI user operation was an **ioctl()**, the repetition count, if any, is that value previously returned or updated in the *ioctlcnt* field.
	MT_RIERRNO	Returns the indication or **errno** value passed in the *tminterrno* fields to the user.
	MT_UPDATEC	Copies all state fields passed in the request and continues the last UPI user operation. **Note:** If the last UPI user operation was an **ioctl()**, the repetition count, if any, is that value previously specified in the *ioctlcnt* field.
	MT_UPDATEE	Copies all state fields passed in the request and returns the indication or **errno** value passed in the *tminterrno* fields to the UPI user.
	MT_UPDATEO	Copies all **tmssi** structure fields into the appropriate state fields in the addressed control block structure.
	MT_UPDATER	Copies all state fields passed in the request and retries the last UPI user operation. **Note:** If the last UPI user operation was an **ioctl()**, the repetition count, if any, is that value previously specified in the *ioctlcnt* field.
tminterrno	**int/errno**	Contains the **int** or **errno** value to be returned to the user if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is RIERRNO or UPDATEE.
upimm		Specifies the address of the logical device that is the subject of this command.
	0	Specifies the logical device image associated with the physical device that corresponds with the *major/minor number* of the device this request addresses that is the subject of this command.
	major/minor	Specifies the *major/minor number* for the logical device that is the subject of this command. If this request is issued against a tape management control point (TMCP), this field must contain the *major/minor number* of the logical device that this request addresses.
ioctlcnt	*nnnn*	Specifies the new **ioctl()** repetition count if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
volumest	*nnnn*	Specifies the new volume state value if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.

	ASGD	Indicates that the volume (device) was assigned to this host. This indication can only occur on a switchable device and means that the last operation issued to this device was an Assign Device **ioctl()** that completed successfully.
	ASGD_ELSE	Indicates that the volume (device) is assigned elsewhere. This indication can only occur on a switchable device and means that the device is not assigned to this host system.
	BOT	Indicates that the volume is at the beginning of tape.
	LEOT	Indicates that the volume is in the logical end of tape space.
	LEOTP	Indicates that the volume is in the logical end of tape space and that this indication was presented to the Tape Manager.
	NOTBPL	Indicates that the volume is ready and not at BOT, PEOT, LEOT, or LEOTP.
	NREADY	Indicates that the volume (device) is not ready.
	PEOT	Indicates that the volume reached the physical end of tape.
	REWIND_VOL	Indicates that the volume is rewinding.
	REWUNL	Indicates that the volume is rewinding and unloading.
	UNLOAD_VOL	Indicates that the volume is unloaded.
	VSUNKNOWN	Indicates that the state of the volume (device) is unknown. This indication can only occur on a switchable device and means that the last operation issued to this device was an Unassign Device **ioctl()** that completed successfully or an Assign Device **ioctl()** that encountered an I/O error.
lastmove		Specifies the new move direction state value if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
	BACKRC	Indicates that the last operation moved the volume in the backward direction and a tape mark was not encountered.
	BACKTM	Indicates that the last operation moved the volume in the backward direction and a tape mark was encountered.
	FORRC	Indicates that the last operation moved the volume in the forward direction and a tape mark was not encountered.
	FORTM	Indicates that the last operation moved the volume in the forward direction and a tape mark was encountered.
	TM2	Indicates that the last operation moved the volume in the forward direction and a second consecutive tape mark was encountered moving in the forward direction.
volpostm	*count*	Specifies the new value that represents the number of tape marks that the volume moved away from the beginning of the tape if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
volposrc	*nnnn*	Specifies the number of records (plus or minus) that the volume moved away from the last tape mark encountered if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
pvpostm	*nnnn*	Specifies the previous number of tape marks that the volume moved away from the beginning of tape if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.

pvposrc	*nnnn*	Specifies the previous number of records (plus or minus) that the volume moved away from the last tape mark encountered if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
phvolpos	*new block ID*	Specifies the new physical block ID value if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is UPDATEC, UPDATEO, UPDATEE, or UPDATER.
timeOut	*secs*	Specifies the number of seconds that this request should wait for an error or exception before returning with an MT_TIMEOUT *ddreason* code.
	0	Specifies the timeout period. The default is 10 minutes.
purgeddr		Controls the purging of a pending *ddreason* code that would normally be presented in response to this request if *subcmd* is PACTIONW, PACTIONWT, SRSIWAIT, or WAITTIMW.
	NO_PURGE	Indicates not to purge any pending *ddreason* code.
	RTRANS	Indicates to purge the pending RTRANS *ddreason* code that would normally be presented with the SRSI request for this device. Subsequent RTRANS *ddreason* codes that occur for this device *will* be presented.

Return Values

On successful completion, the *ddreason* field contains a reason code that reflects the condition that caused the device driver to return to the requestor. Except as noted in the individual reason code descriptions, the UPI process that set the *ddreason* field is suspended until a subsequent **ioctl(...MTTMSSRSI...)** request is received that specifies a PACTION subcommand. The *pwait* field indicates whether the process that set the reason code is suspended waiting for tape management action to continue.

The fields contain the following values when this system call completes successfully:

resvd	**0**	Reserved.
versn	**TMS_SRSIVSN**	Specifies the block structure version number.
ddreason		Specifies the reason code explained previously. The following legend is used to identify the process wait condition for each *ddreason* code: *[w/nw]*, where *w* = process wait condition and *nw* = no process wait condition.
	ACLA	Indicates that the automatic cartridge loader feature for this device went from an inactive state to an active state. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*
	BOT	Indicates that the beginning of tape was encountered as the result of a logical device operation. *[w]*
	DDBSB	Indicates that the user issued a backward space block (BSB) **ioctl()**. *[w]*
	DDBSBTM	Indicates that the user issued a backward space block (tape mark) MTTMSBLK **ioctl()**.
	DDBSF	Indicates that the user issued a backward space file (BSF) **ioctl()**. *[w]*

DDFSB Indicates that the user issued a forward space block (FSB) **ioctl()**. *[w]*

DDFSBTM Indicates that the user issued a forward space block (tape mark) MTTMSBLK **ioctl()**. *[w]*

DDFSF Indicates that the user issued a forward space file (FSF) **ioctl()**. *[w]*

DDREW Indicates that the user issued a rewind (REW) **ioctl()**. *[w]*

DDUNL Indicates that the user issued an (rewind and) unload (UNL) **ioctl()**. *[w]*

DDWTM Indicates that the user issued a write tape mark (WTM) **ioctl()**. *[w]*

ERRORI Indicates that a nonspecific error condition was returned from the device while the user was doing an **ioctl()** operation. *[w]*

ERRORR Indicates that a nonspecific error condition was returned from the device while the user was doing a read operation. *[w]*

ERRORW Indicates that a nonspecific error condition was returned from the device while the user was doing a write operation. *[w]*

FILE_PROT Indicates that the user issued a write operation to a volume that is write protected. *[w]*

IFORM Indicates that the user issued a format change **ioctl()** request. The recording format requested can be obtained using the Set/Read Mode Information (SRMI) **ioctl()**. *[w]*

INT_ERROR Indicates that an internal device driver error (for example, a bad pointer) was detected for this MPI/UPI device. *[w]*

ISTTM Indicates that the user executed a backspace **ioctl()** operation that encountered the first tape mark on the volume. *[w]*

LEOT Indicates that the logical end of tape space was encountered as the result of a logical device operation. *[w]*

MPICLOSE Indicates that a **close()** request was received for an MPI device that still has a logical device (UPI) associated with the MPI device. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*

MT_CONF Indicates that a configuration change or a configuration change attempt occurred. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*

MT_ERROR Indicates that an error condition was returned from the device. *[w]*

MT_TIMEOUT Indicates that the timeout period as specified by a PACTIONWT or WAITTIME *subcmd* expired. *[nw]*

NO_WAIT Indicates the normal completion of one of the “no-wait” commands as defined in the *subcmd* field. *[nw]*

NREADY Indicates that the user issued a system call for this logical device and the device is not ready or has gone not ready. *[w]*

OPEN2 Indicates that another **open()** was attempted against this MPI device that was not through the logical device interface. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*

PEOT Indicates that the physical end of tape was encountered as the result of a logical device operation. *[w]*

	RDNTFY	Indicates that the user issued a **read()** for this logical device and the *rdrtn* mode setting in the previous Set/Read Mode Information (SRMI) **ioctl()** system call specified "ALWAYS." *[w]*
	REVENT	Indicates that a reset event condition was detected for this device. *[w]*
	RTRANS	Indicates that a ready transition condition was detected for this device. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*
	SRLF	Indicates that the user issued an **srlf() ioctl()** for this logical device. *[w]*
	STATI	Indicates that a statistical counter maintained for this device by the device driver overflowed. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process. *[nw]*
	TM	Indicates that a tape mark was detected by a logical device operation. *[w]*
	TM2	Indicates that a second consecutive tape mark was encountered going in the forward direction by a logical device operation. *[w]*
	UPIC	Indicates that the user issued an **close()** for this logical device. *[w]*
	UPINA	Indicates that the user issued a system call for this logical device and the "NOTAVAIL" mode was specified in the previous Set/Read Mode Information **ioctl()** (SRMI). *[w]*
	UPIO	Indicates that the user issued an **open()** for this logical device and the "openrtn" mode setting in the previous Set/Read Mode Information **ioctl()** (SRMI) specified the value "ALWAYS." *[w]*
	WEV	Indicates that the user issued an **open()** for this logical device, the "rwmode" mode setting in the previous Set/Read Mode Information **ioctl()** (SRMI) specified the value "ROWNE," and the volume has write capability enabled. *[w]*
	WRTNBS	Indicates that the count passed with a write system call for this User Protocol Interface device was not equal to, or if allowable, a multiple of the "block size" value set by the Set/Read Mode Information system call for this user. *[w]*
	WRTNTFY	Indicates that the user issued a **write()** for this logical device and the "wrtrtn" mode setting in the previous Set/Read Mode Information **ioctl()** (SRMI) specified the value "ALWAYS." *[w]*
pwait		Indicates a wait or suspended status for the process that presented a *ddreason* code that is being returned or was returned to the caller of this request.
	MT_NOWAIT	Indicates that there are currently no processes suspended waiting for tape management action (*ddaction*) that have presented notification through the *ddreason* code.
	MT_WAIT	Indicates that a process requiring tape management action was presented notification through the *ddreason* code and is suspended waiting for tape management action (*ddaction*).
pwaitp		Indicates the wait status for a process that posted a *ddreason* code, but has not yet presented that *ddreason* code to the caller of the request.
	MT_NOWAIT	Indicates that there are currently no processes suspended waiting to present *ddreason* code notification to the caller of this request.

	MT_WAIT	Indicates that a process requiring tape management support (*ddaction*) is suspended waiting to present *ddreason* code notification to the caller of this request.
ddrcnt	*nnnn*	Specifies the number of *ddreason* codes waiting to be presented.
upimm	*major/minor number*	Specifies the *major/minor number* of the logical device that the state information represents.
ddinterrno	**int/error**	Indicates that if the *ddreason* field contains a reason code showing that an error or exception condition detected by the device driver, this field contains the **int** or **errno** value that the device driver would have returned to the user if this same condition was encountered on a device that is not under the control of a tape management system and not designated as a logical device.
ioctlcnt	*nnnn*	Specifies the repetition count value, if any, if an **ioctl()** is being executed.
curscst	*current call*	Specifies the current logical device system call being executed (or requested to be executed).
	BSB_ST	Indicates the backspace block.
	BSBTM_ST	Indicates the backspace block (tape mark).
	BSF_ST	Indicates the backspace file.
	CLOSE_ST	Indicates a close.
	DSE_ST	Indicates a data security erase.
	ERG_ST	Indicates an erase record gap.
	FORM_ST	Indicates a format change.
	FSB_ST	Indicates a forward space block.
	FSBTM_ST	Indicates a forward space block (tape mark).
	FSF_ST	Indicates a forward space file.
	IOCTL_ST	Indicates that an **ioctl()** was executed. The specific command was not evaluated.
	LBID_ST	Indicates a locate block ID.
	NONE_ST	Indicates that a system call is not currently being executed.
	OPEN_ST	Indicates an open.
	OTH_ST	Indicates a valid, nondefined **ioctl()**.
	RBID_ST	Indicates a read block ID.
	RBLOG_ST	Indicates a read buffer log.
	READ_ST	Indicates a read.
	REW_ST	Indicates a rewind.
	SRLF_ST	Specifies a set/read last failure indication.
	UNL_ST	Indicates an unload.
	WRT_ST	Indicates a write.
	WTM_ST	Indicates a write tape mark.
priscst	*prior call*	Specifies the prior system call executed by the logical device. Refer to the *curscst* field for a definition of the values that can be returned.
volumest		Specifies the volume state.

	ASGD	Indicates that the volume (device) was assigned to this host. This indication can only occur on a switchable device and means that the last operation issued to this device was an Assign Device **ioctl()** that completed successfully.
	ASGD_ELSE	Indicates that the volume (device) is assigned elsewhere. This indication can only occur on a switchable device and means that the device is not assigned to this host system.
	BOT	Indicates that the volume is at the beginning of tape.
	LEOT	Indicates that the volume is in the logical end of tape space.
	LEOTP	Indicates that the volume is in the logical end of tape space and this indication was presented to the Tape Manager.
	NOTBPL	Indicates that the volume is ready and not at BOT, PEOT, LEOT, or LEOTP.
	NREADY	Indicates that the volume (device) is not ready.
	PEOT	Indicates that the volume reached the physical end of tape.
	REWIND_VOL	Indicates that the volume is rewinding.
	REWUNL	Indicates that the volume is rewinding and unloading.
	UNLOAD_VOL	Indicates that the volume is unloaded.
	VSUNKNOWN	Indicates that the state of the volume (device) is unknown. This indication can only occur on a switchable device and means that the last operation issued to this device was an Unassign Device **ioctl()** that completed successfully or an Assign Device **ioctl()** that encountered an I/O error.
lastmove		Indicates that this field contains the current move direction state value.
	BACKRC	Indicates that the last operation moved the volume in the backward direction and a tape mark was not encountered.
	BACKTM	Indicates that the last operation moved the volume in the backward direction and a tape mark was encountered.
	FORRC	Indicates that the last operation moved the volume in the forward direction and a tape mark was not encountered.
	FORTM	Indicates that the last operation moved the volume in the forward direction and a tape mark was encountered.
	TM2	Indicates that the last operation moved the volume in the forward direction and that a second consecutive tape mark was encountered moving in the forward direction.
volpostm	*nnnn*	Specifies the number of tape marks that the volume moved away from the beginning of the tape. If an error condition is being reported, the *volpostm* field and the *volposrc* field represent the tape position prior to the operation that failed. If a nonerror condition (for example, a tape mark or logical end of tape) is being reported, the *volpostm* field and the *volposrc* field reflect the actual tape mark and record count.

volposrc	*nnnn*	Specifies the number of records (plus or minus) that the volume moved away from the last tape mark encountered. If an error condition is being reported, the *volpostm* field and the *volposrc* field represent the tape position prior to the operation that failed. If a nonerror condition (for example, a tape mark or logical end of tape) is being reported, the *volpostm* field and the *volposrc* field reflect the actual tape mark and record count.
pvpostm	*nnnn*	Specifies the previous number of tape marks that the volume moved away from the beginning of tape.
pvposrc	*nnnn*	Specifies the previous number of records (plus or minus) that the volume moved away from the last tape mark encountered.
phvolpos	*block ID*	Indicates either the last physical block ID value that a successful locate physical block command referenced, or the last physical block ID that was returned when a RDSENSE subcommand was issued to a 3480 or 3490 device.
	X'7FFFFFFF'	Indicates that the physical block ID is not valid.
mtoflags	**O_flags**	Specifies the O_Flags passed by the last UPI **open()** system call. If the UPI device is not open or is not currently processing an **open()** system call, this field contains -1.
siflags		Specifies a bit-significant field of miscellaneous state information.
	MTPOS_UNRELIABLE	Indicates (0x80) that a locate block ID **ioctl()** was successfully executed by the UPI user. The tape mark and block count values (*volpostm*, *volposrc*, *pvpostm*, and *pvposrc*) maintained by the device driver may no longer be valid. **Note:** This state indication is reset upon presentation. (0x40–0x01) Reserved.
purgeddr	**NO_PURGE**	Returns when the *subcmd* field is RDSTATE.
	nnnn	Specifies the value that was passed with this SRSI **ioctl()** request when the *subcmd* field is *not* RDSTATE.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY]	Indicates that this request was issued to an MPI device that has a previously issued SRSI request outstanding.
[EINTR]	Indicates that the request was interrupted by a signal.
[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[EINVAL]	Indicates that the subcommand field value is undefined.
[EINVAL]	Indicates that the command was issued to a standard device, a UPI device, or a TMPCP.
[EINVAL]	Indicates that an update command attempted to set a field to an invalid value.

[EINVAL]	Indicates that the subcommand is PACTIONR, PACTIONW, or PACTIONWT and that the *ddaction* is invalid.
[EINVAL]	Indicates that an attempt was made to send an action code to a UPI that is not waiting for an action code.
[EINVAL]	Indicates that the security key is not valid.
[EIO]	Indicates that the SENSE channel command failed on a RDSENSE command.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.
[ERESTART]	Restarts the current request.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mt_srsti

Purpose

mt_srsti – Sets, changes, or reads tape device statistical information.

Note: This is included for completeness. Refer to the tape device driver product commands for the latest interface definition.

Syntax

#include <**sys/mttms.h**>

int ioctl(*fildes***,MTTMSSRSTI,***arg***);**
int fildes,MTTMSSRSTI;
struct tmssti ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSSRSTI	Specifies the **ioctl()** request to set, change, or read statistical information for a magnetic tape device.
arg	Specifies the address of the **tmssti** structure passed.

Description

This **ioctl()** system call sets or reads statistical information associated with a logical device image created by a previous MTTMSCUPI **ioctl()** system call. This logical device image is referred to as the User Protocol Interface (UPI) in the documentation.

This **ioctl()** system call can only be issued against a device that was opened in Management Protocol Interface (MPI) mode. It cannot be issued against a tape management pseudo control point (TMPCP).

The **tmssti** structure is defined in **mttms.h** as follows:

```
struct tmssti {
int    resvd;        /* Reserved                                    */
int    versn;        /* Version number field                        */
int    secukey;      /* Security key                                */
char   subcmd;       /* Subcommand field                            */
dev_t  upimm;        /* Major/minor number of the logical device    */
int    volbyterd;    /* Number of bytes read since volume mounted   */
int    volbytewt;    /* Number of bytes written since volume mounted */
double devbyterd;    /* Number of bytes read from the device        */
double devbytewt;    /* Number of bytes written to the device       */
int    volread;      /* Number of reads since volume mount          */
int    volwrt;       /* Number of writes since volume mount         */
int    voloth;       /* Number of other moves since volume mount    */
int    devread;      /* Number of reads issued to device            */
int    devwrt;       /* Number of writes issued to device           */
int    devoth;       /* Number of other movements issued to device  */
int    volrerr;      /* Number of read errors for volume            */
int    volwerr;      /* Number of write errors for volume           */
int    voloerr;      /* Number of other errors for volume           */
int    devrerr;      /* Number of read errors for device            */
int    devwerr;      /* Number of write errors for device           */
int    devoerr;      /* Number of other errors for device           */
};
```

For a request to read statistical information, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_SRSTIVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSK **ioctl()**.
subcmd	**RDMODE**	Returns the current mode information for this User Protocol Interface device.
	RDSTAT	Returns the current statistical information for this device.

For a request to write statistical information, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_SRSTIVSN**	Specifies the block structure version number.
subcmd	**WRTSTAT**	Initializes the statistical information records for this device to the value contained in the following fields.
upimm		Specifies the *major/minor number* of the UPI device.
	0	Specifies the logical device image associated with the physical device that corresponds with the *major/minor number* of the device this request addresses that is the subject of this command. If this request is issued against a tape management control point (TMCP) entry, this field must contain the *major/minor number* of the logical device that this command is issued against.
	major/minor number	Specifies the *major/minor number* of the device.
volbyterd	*nnnn*	Specifies the new number of bytes read from this volume since the last SRSTI **ioctl()** was issued.

volbytewt	*nnnn*	Specifies the new number of bytes written to this volume since the last SRSTI **ioctl()** was issued.
devbyterd	*nnnn*	Specifies the number of bytes read from this device since the last SRSTI **ioctl()** was issued with the WRTSTAT *subcmd*.
devbytewt	*nnnn*	Specifies the number of bytes written to this device since the last SRSTI **ioctl()** was issued with the WRTSTAT *subcmd*.
volread	*nnnn*	Specifies the new count of read operations that were issued against this volume since the last SRSTI **ioctl()** was issued.
volwrt	*nnnn*	Specifies the new count of write operations that were issued against this volume since the last SRSTI **ioctl()** was issued.
voloth	*nnnn*	Specifies the count of other operations that were issued against this volume since the last SRSTI **ioctl()**.
devread	*nnnn*	Specifies the new count of read operations that were issued since the last time this value was set by an SRSTI **ioctl()**.
devwrt	*nnnn*	Specifies the new count of write operations that were issued against this device since the last time this value was set by an SRSTI **ioctl()**.
devoth	*nnnn*	Specifies the new count of other movement operations that were issued against this device since the last time this value was set by an SRSTI **ioctl()**.
volrerr	*nnnn*	Specifies the new count of read errors accumulated since the last reset by an SRSTI **ioctl()**.
volwerr	*nnnn*	Specifies the new count of write errors accumulated since the last reset by an SRSTI **ioctl()**.
voloerr	*nnnn*	Specifies the new count of other errors accumulated since the last reset by an SRSTI **ioctl()**.
devrerr	*nnnn*	Specifies the new count of read errors accumulated against this device since the last reset by an SRSTI **ioctl()**.
devwerr	*nnnn*	Specifies the new count of write errors accumulated against this device since the last reset by an SRSTI **ioctl()**.
devoerr	*nnnn*	Specifies the new count of other errors accumulated against this device since the last reset by an SRSTI **ioctl()**.

Return Values

On successful completion, when the *subcmd* field contains RDSTAT, the following fields are returned in the structure:

resvd	**0**	Reserved.
versn	**TMS_SRSTIVSN**	Specifies the block structure version number.
subcmd	**RDSTAT**	Returns the current statistical information for this device and resets the *volread*, *volwrt*, and *voloth* fields.
upimm		Specifies the *major/minor number* of the UPI device.
volbyterd	*nnnn*	Indicates either the number of bytes read from this volume since the last SRSTI command issued with the RDSTAT subcommand, or the difference between the last value specified with the WRTSTAT command plus the number of bytes read from this volume since the last SRSTI operation.

volbytewt	*nnnn*	Indicates either the number of bytes written to this volume since the last SRSTI command issued with the RDSTAT subcommand, or the difference between the last value specified with the WRTSTAT command plus the number of bytes written to this volume since the last SRSTI operation.
devbyterd	*nnnn*	Indicates the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of **read()** operations that were issued against this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
devbytewt	*nnnn*	Indicates the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of **write()** operations that were issued against this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
volread	*nnnn*	Specifies either the number of **read()** operations that were issued against this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of **read()** operations that were executed against this volume since the last SRSTI operation.
volwrt	*nnnn*	Specifies either the number of **write()** operations that were issued against this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of **write()** operation that were executed against this volume since the last SRSTI operation.
voloth	*nnnn*	Specifies either the number of **ioctl()** operations that were issued against this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of **ioctl()** operation that were executed against this volume since the last SRSTI operation.
devread	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of **read()** operations that were issued against this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
devwrt	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of **write()** operations that were issued against this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
devoth	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of other movement operations that were issued against this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
volrerr	*nnnn*	Specifies either the number of read errors that were encountered by this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of read errors that were encountered by this volume since the last SRSTI operation.
volwerr	*nnnn*	Specifies either the number of write errors that were encountered by this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of write errors that were encountered by this volume since the last SRSTI operation.
voloerr	*nnnn*	Specifies either the number of other errors that were encountered by this volume since the last SRSTI command was issued with a *subcmd* of RDSTAT, or the incremental difference of the last value specified with a *subcmd* of WRTSTAT plus the number of other errors that were encountered by this volume since the last SRSTI operation.

devrerr	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of read errors that were encountered by this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
devwerr	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of write errors that were encountered by this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.
devoerr	*nnnn*	Specifies the incremental difference of the last value specified with a WRTSTAT *subcmd*, plus the number of other errors that were encountered by this device since the last time this value was set by a SRSTI **ioctl()** that specified the WRTSTAT *subcmd*.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[EINVAL]	Indicates that the subcommand field is invalid.
[EINVAL]	Indicates that the command was issued to a standard device, a UPI device, or a TMPCP.
[EINVAL]	Indicates that the command was issued to a TMCP and the *upimm* field is 0.
[EINVAL]	Indicates that the security key is not valid.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

Related Information

Refer to the **mt_tmcp** command for more information on tape management enablement logic for tape device drivers. Also refer to the related **ioctls()** for magnetic tape functions.

mte_cskey

Purpose

mte_cskey – Establishes a system security key in the pseudo tape device driver for tape management **ioctl()** access. This security key is required on systems that do not support the “tmsmpi” C2 security privilege.

Syntax

#include <**sys/mte_tms.h**>

int ioctl(*fildes***,MTTMSCSKE,***arg***);**
int *fildes***, MTTMSCSKE;**
struct cskeye ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSCSKE	Specifies the **ioctl()** request to create a system security key.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call establishes a system security key required for Management Protocol Interface (MPI) **ioctl()** access to the MTE pseudo tape device driver on systems that do not support the “tmsmpi” C2 security privilege. It can only be issued against a tape management pseudo control point (TMPCP) and must be issued immediately following the opening of the TMPCP.

The **cskeye** structure is defined as follows:

```
struct cskeye {
int  resvd;       /* Reserved                                    */
int  versn;       /* Version number field                        */
int  cskstatus;   /* Security key status                         */
int  secukey;     /* Security key                                */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_CSKEVSN**	Specifies the block structure version number.
secukey		Specifies the system security key to be established for Management Protocol Interface (MPI) **ioctl()** access. This can be any nonzero value.

Return Values

On successful completion, the following values are returned in the **cskeye** structure:

resvd	**0**	Reserved.
versn	**TMS_CSKEVSN**	Specifies the block structure version number passed with the request.
cskstatus		Contains the security key status.
	SK_C2	Indicates that the operating system supports the "tmsmpi" C2 Security privilege, therefore, the security key passed in the *secukey* field was not accepted.
	SK_EXISTS	Indicates that a security key was already created for this TMPCP. The security key passed in the *secukey* field was not accepted.
	SK_VALID	Indicates that the security key passed in the *secukey* field was established as a valid security key.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY]	Indicates that someone else is using the device specified in *fildes*.
[EINVAL]	Indicates that either an invalid field was detected on the request, or that the *major/minor number* for this request did not have the MPI bit set.
[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.
[EPERM]	Indicates that the operating system supports the "tmsmpi" C2 privilege and that the requesting process does not have the "tmsmpi" privilege.
[ ]	A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_cupi

Purpose

mte_cupi – Creates a user access path to a logical device image in the pseudo tape device driver.

Syntax

#include <**sys**/**mte_tms.h**>
int ioctl(*fildes***,MTTMSCUPIE,***arg***);**
int *fildes***, MTTMSCUPIE;**
struct cupie ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSCUPIE	Specifies the **ioctl()** request to create a user access path logical device entry within the MTE pseudo tape device driver.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call creates a logical device image in an MTE pseudo tape device driver. The logical device created represents an image of the physical device to which user requests can be issued. This image is referred to as the User Protocol Interface (UPI). Requests made to the UPI logical device image must have the UPI bit set in the device minor number.

This **ioctl()** system call returns the content of the device major/minor numbers (with the UPI bit set) that can be used to create device special files and subsequently used to access the logical device image in UPI mode. It can only be issued against a device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device.

The **cupie** structure is defined as follows:

```
struct cupie {
int   resvd;       /* Reserved                                        */
int   versn;       /* Version number field                            */
int   secukey;     /* Security key                                    */
int   cupiopt;     /* Options field, for future use                   */
ulong cupistatus;  /* Request completion status                       */
dev_t devmmn;      /* Device major/minor number                       */
dev_t cupimm1;     /* Major/minor number assigned to this entry       */
dev_t cupimm2;     /* Major/minor number assigned to this entry       */
dev_t cupimm3;     /* Major/minor number assigned to this entry       */
char  cupitype;    /* Logical device image type                       */
char  cupimode;    /* Connection mode                                 */
int   cupideva;    /* Physical device address                         */
ulong cupiipa;     /* IP address for this logical device image        */
ulong cupiport;    /* Port address for this logical device image      */
ulong cupibuff;    /* Def. data buffer size for this log. dev. image  */
int   usrspaceln;  /* Length of the user space                        */
char  *usrspace;   /* User space address                              */
};
```

On request, the fields contain:

resvd **0**
Reserved.

versn **TMS_CUPIEVSN**
Specifies the block structure version number.

secukey
Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSKE **ioctl()**.

cupiopt
Reserved.

devmmn *major/minor number*
Indicates the *major/minor number* for the device that this UPI logical device image is being created for. This entry is required for all logical device image types except STUB.

Usage notes:

2 For requests issued against a TMPCP with a *cupitype* of MAGTAPE or CP_LOCAL, the *major/minor number* value contained in this field is normally obtained from the *cupimm1* field. The *cupimm1* field is returned from the corresponding MTTMSCUPI **ioctl()** command issued to an MPI device or TMCP on the same host system.

3 For requests issued against a TMPCP with a *cupitype* of CP_REMOTE, the *major/minor number* value contained in this field is normally obtained from the *cupimm1* field. The *cupimm1* field is returned from the corresponding MTTMSCUPIE **ioctl()** command issued to the TMPCP to create the local connection point (*cupitype* = CP_LOCAL) on the host that contains the target device.

cupitype
Specifies the logical device image type to be created. The defined entries include:

CP_LOCAL
Specifies the logical device image for a local connection point. The *major/minor number* of the device is passed in the *devmmn* field.

Note: Any UPI user request received for a logical device image of CP_LOCAL is returned with an error.

	CP_REMOTE	Specifies the logical device image for a remote connection point. The *major/minor number* passed in the *devmmn* field is the *major/minor number* of the tape device (on the CP_LOCAL node) that is to be accessed through this remote connection point.
	MAGTAPE	Specifies the logical device image for a magnetic tape device. The *major/minor number* of the device is passed in the *devmmn* field.
	STUB	Specifies the logical device image for a null device. **Note:** Any UPI user request received for a logical device image of STUB is returned with an **errno** value of ENXIO.
cupimode		Specifies the connection mode for this logical device image. The defined entries include:
	MTE_NORMAL	Indicates that this logical device image functions in a normal manner. TMPCP notification and UPI request queueing does not occur.
	MTE_NOTIFY	Indicates that when a UPI request is received for this logical device image, the request is queued and notification is provided through the normal SRSIE **ioctl()** system call mechanism, via the TMPCP for this pseudo tape device/logical device image.
cupideva	*address*	Specifies the physical device address.
cupiipa	*address*	Indicates that when *cupitype* is CP_LOCAL, this field contains the IP address that is to be used for socket creation for this local logical device image. When *cupitype* is CP_REMOTE, this field contains the IP address that was used to create the socket interface for the CP_LOCAL logical device image. The *cupiipa* along with the *cupiport* represent the address to be used to establish a socket connection path between this CP_REMOTE logical device image and the target CP_LOCAL logical device image.
cupiport	*address*	Indicates that when *cupitype* is CP_REMOTE, this field contains the port address for the logical device image that was returned when the CP_LOCAL logical device image was created. The *cupiipa* along with *cupiport* represent the address to be used to establish a socket connection path between this CP_REMOTE logical device image and the target CP_LOCAL logical device image.
cupibuff	*length*	Indicates that when *cupitype* is CP_LOCAL, this field contains the data buffer size that should be allocated (initially) for data transfer operations from or to the physical tape device driver. If this field is 0, the default data buffer size is 64KB.
usrspaceln	*length*	Indicates that when *cupitype* is CP_LOCAL, this field contains the length of the user space passed in the *usrspace* field.
usrspace	*address*	Indicates that when *cupitype* is CP_LOCAL, this field contains an address in user space that was reserved by the caller of this request for use by the CP_LOCAL logical device image. This space is used by the CP_LOCAL logical device image to map “ill-behaved” **ioctl()** system calls and, therefore, it must remain for the duration of the existence of the CP_LOCAL logical device image. The size of the user space is passed in the *usrspaceln* field and needs to be equal to or larger than the maximum size **ioctl()** argument supported for the tape device represented by this CP_LOCAL logical device image.

Return Values

The fields contain the following values when this system call completes successfully.

On return, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_CUPIEVSN**	Specifies the block structure version number passed with the request.
cupistatus	*request completion status*	Indicates the request completion status for the creation of this logical device image. The defined entries include:
	REQ_COMP	Indicates that the request to create this logical device image is complete. The logical device image is available for use.
	REQ_IN_PROCESS	Indicates that the request to create this logical device image is in process, but not yet complete. The logical device image is not available for use at this time. Completion status for this logical device image creation is reported through the normal SRSIE **ioctl()** system call mechanism, via the TMPCP for this pseudo tape device/logical device image.
cupimm1	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* when used to issue a user (UPI) request is *no rewind* and *no unload*.
cupimm2	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* when used to issue a user (UPI) request is *rewind*.
cupimm3	*major/minor number*	Contains the UPI *major/minor number* assigned by the device driver for this logical device image. The **close()** disposition for this *major/minor number* when used to issue a user (UPI) request is *rewind* and *unload*.
cupiport	*address*	Indicates that when *cupitype* is CP_LOCAL, this field contains the port address for the socket interface that was created for this local logical device image.

A return value of 0 indicates a successful operation. A return value of -1 indicates the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY] Indicates that someone else is using the device specified in *fildes*.

[EINVAL] Indicates that the version field is out of date. Recompile the application.

[EINVAL] Indicates that either a security key mismatch was detected, or that the security key was not established.

[EINVAL] Indicates that the **ioctl()** request was issued without the MPI bit set in the device minor number.

[EINVAL] Indicates an invalid *devmmn*, *cupitype*, or *cupimode* for the device.

[ENOMEM] Indicates that no memory is available for the UPI control block.

[ENXIO] Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

[ ] Indicates that a socket interface was not created for a CP_LOCAL or CP_REMOTE logical device image. The **errno** value from the function that failed is returned.

[ ] A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_eupi

Purpose

mte_eupi – Inserts or extracts user data or **ioctl()** parameters to or from a logical device image in the pseudo tape device driver.

Syntax

#include <**sys**/**mte_tms.h**>
int ioctl(*fildes***,MTTMSEUPIE,***arg***);**
int *fildes***, MTTMSEUPIE;**
struct eupie ****arg***;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSEUPIE	Specifies the **ioctl()** request used to insert or extract user data or **ioctl()** parameters to or from a logical device image in the pseudo tape device driver.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call inserts or extracts user data or **ioctl()** parameters to or from a logical device image in the pseudo tape device driver. It can only be issued against a tape management pseudo control point (TMPCP) and must reference a logical device image created by a previous create UPI **mte_cupi ioctl()** system call.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a TMPCP.

This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device.

The **eupie** structure is defined as follows:

```
struct eupie {
int   resvd;        /* Reserved                                        */
int   versn;        /* Version number field                            */
int   secukey;      /* Security key                                    */
dev_t upimm;        /* Major/minor number of logical device image      */
char  subcmd;       /* Subcommand field                                */
char  status:       /* Operation status                                */
char  *buffaddr;    /* Address of data/parm buffer                     */
int   bufflngth;    /* Length of data/parm buffer                      */
int   bytesxferd;   /* Bytes transferred to or from data/parm buffer   */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_EUPIEVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSKE **ioctl()**.
upimm	*major/minor*	Contains the *major/minor number* for the logical device image (associated with this TMPCP) that is the subject of this command.
subcmd	**EDATA**	Extracts (copies) data from the user request data buffer for this logical device image to the data buffer address passed in *buffaddr*.
	IDATA	Inserts (copies) data from the data buffer address passed in *buffaddr* to the IDATA buffer allocated for this logical device image.
	EPARM	Extracts (copies) data from the user **ioctl()** request parameter buffer for this logical device image to the data buffer address passed in *buffaddr*.
	IPARM	Inserts (copies) data from the data buffer address passed in *buffaddr* to the IPARM buffer allocated for this logical device image.
buffaddr		Contains the address of the buffer to be used for the requested data operation.
bufflngth	*nnnn*	Specifies the length of the buffer identified in the *buffaddr* field.

Return Values

On successful completion, the *status* field contains a completion status for the requested operation.

status		Contains the status for the operation requested in the *subcmd* field.
	SUCCESS	Indicates that the operation completed successfully.
	INVALID	Indicates that the *subcmd* did not contain a valid request.
	LENGTH	Indicates that if the *subcmd* is IDATA, the bytes specified in the *bufflngth* field exceeded the size of the logical image user data buffer. The operation is not performed. Indicates that if the *subcmd* is IPARM, the bytes specified in the *bufflngth* field is not equal to the logical image **ioctl()** parameter buffer length. The operation is not performed. Indicates that if the *subcmd* is EDATA or EPARM, the bytes specified in the *bufflngth* field is less than the size of the logical image user data buffer or **ioctl()** parameter buffer length. The operation is not performed.

	NO_BUFF	Indicates that the *subcmd* was EDATA and this is not a CP_LOCAL logical device image, or this CP_LOCAL logical device image does not have valid data returned from the current or previous data request operation (this response is also returned when a successful read operation transferred zero bytes). Indicates that the *subcmd* was IDATA and this is not a CP_REMOTE logical device image, or this CP_REMOTE logical device image process is not blocked with an outstanding read-type system call pending tape manager action. Indicates that the *subcmd* was EPARM and this is not a CP_LOCAL logical device image, or this CP_LOCAL logical device image does not have valid **ioctl()** parameter data returned from the current or previous data request operation. Indicates that the *subcmd* was IPARM and this is not a CP_REMOTE logical device image, or this CP_REMOTE logical device image process is not blocked with an outstanding **ioctl()** system call pending tape manager action.
bytesxferd	*nnnn*	Indicates that when *status* is SUCCESS, this field contains the actual number of bytes transferred to or from the data buffer specified by *buffaddr*.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY]	Indicates that this request was issued to a TMPCP that has a previously issued request outstanding.
[EINTR]	Indicates that the request was interrupted by a signal.
[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[EINVAL]	Indicates that a *subcmd* field value was not valid.
[EINVAL]	Indicates that the command was issued to a device other than a TMPCP.
[EINVAL]	Indicates that the *upimm* field does not point to a valid logical device image.
[EINVAL]	Indicates that a security key mismatch was detected or that the security key was not established.
[EINVAL]	Indicates that this **ioctl()** request was issued without the MPI bit set in the device minor number.
[ENOMEM]	Indicates that the data buffer needed to satisfy an IDATA request was unavailable.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing any further device activity.

[ ]	A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_mupi

Purpose

mte_mupi – Modifies a user access path to a logical device image in the pseudo tape device driver.

Syntax

#include <**sys**/**mte_tms.h**>
int ioctl(*fildes***,MTTMSMUPIE,***arg***);**
int *fildes***, MTTMSMUPIE;**
struct mupie ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSMUPIE	Specifies the **ioctl()** request used to modify a user access path logical device image within the pseudo device driver.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call modifies a UPI logical device image previously created by a MTTMSCUPIE **ioctl()** system call. It can only be issued against a tape management pseudo control point (TMPCP) and must reference a logical device image created by a previous create UPI **mte_cupi ioctl()** system call.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a TMPCP.

This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device.

The **mupie** structure is defined as follows:

On request, the fields contain:

```
struct mupie {
int    resvd;         /* Reserved                                    */
int    versn;         /* Version number field                        */
int    secukey;       /* Security key                                */
ulong  mupistatus;    /* Request completion status                   */
dev_t  mupimm1;       /* Major/minor number assigned to this entry   */
dev_t  devmmn;        /* Device major/minor number                   */
char   mupitype;      /* Logical device image type                   */
char   mupimode;      /* Connection mode                             */
int    mupideva;      /* Physical device address                     */
ulong  mupiipa;       /* IP address for this logical device image    */
ulong  mupiport;      /* Port address for this logical device image  */
ulong  mupibuff;      /* Def. data buffer size for this log dev image */
ulong  mupireqst;     /* State of user (UPI) req for this log dev img */
int    usrspaceln;    /* Length of the user space                    */
char   *usrspace;     /* User space address                          */
char   curscst;       /* Current system call state                   */
int    next_state;    /* Next logical image called - state           */
int    next_interrno; /* Next log. image called - int/errno returned */
int    next_rw_cnt;   /* Next logical image called - read/write cnt  */
int    next_ioctl;    /* Next logical image called - ioctl type      */
};
```

resvd **0** Reserved.

versn **TMS_MUPIEVSN** Specifies the block structure version number.

secukey Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key returned with the MTTMSCSKE **ioctl()**.

mupimm1 Specifies the *cupimm1 major/minor number* that was returned by the MTTMSCUPIE **ioctl()** system call when this logical device image was created.

devmmn Specifies the *major/minor number* assigned to the device represented by this logical device image.

0 Indicates not to modify the *major/minor number* for this device.

major/minor number Specifies the new *major/minor number* for this device.

Usage notes:

2 For requests issued against a TMPCP with a *cupitype* of MAGTAPE or CP_LOCAL, the *major/minor number* value contained in this field is normally obtained from the *cupimm1* field returned from the corresponding MTTMSCUPI **ioctl()** command issued to an MPI device or TMCP on the same host system.

3 For requests issued against a TMPCP with a *cupitype* of CP_REMOTE, the *major/minor number* value contained in this field is normally obtained from the *cupimm1* field returned from the corresponding MTTMSCUPIE **ioctl()** command issued to the TMPCP to create the local connection point (*cupitype* = CP_LOCAL) on the host that contains the target device.

mupitype Specifies the logical device image type. The defined entries include:

0 Indicates not to modify the type for this logical device image.

CP_LOCAL Changes this logical device image type to a local connection point.

Note: Any UPI user request received for a logical device image of CP_LOCAL is returned with an error.

	CP_REMOTE	Changes this logical device image type to a remote connection point. **Note:** Refer to the **mte_cupi** command for more information on required fields when changing from one logical device image type to another logical device image type.
	MAGTAPE	Changes this logical device image type to a magnetic tape device entry.
	STUB	Changes this logical device image type to a null device. **Note:** Any UPI user request received for a logical device image of STUB is returned with an **errno** value of ENXIO.
mupimode		Specifies the connection mode for this logical device image. The defined entries include:
	0	Indicates not to modify the connection mode for this logical device image.
	MTE_NORMAL	Places this logical device image in MTE_NORMAL mode. TMPCP notification and UPI request queuing does not occur.
	MTE_NOTIFY	Places this logical device image in MTE_NOTIFY mode. When a UPI request is received for this logical device image, the request will be queued and notification will be provided through the normal SRSIE **ioctl()** system call mechanism (by TMPCP) for this pseudo tape device driver/logical device image.
mupideva		Specifies the device address for this logical device image.
	0	Indicates not to modify the physical device address for this logical device image.
	physical device address	Specifies the new physical device address to be used for this logical device image.
mupiipa		Specifies the IP address for the local device image. This field is only valid if this request is issued for a new or existing device type of CP_LOCAL or CP_REMOTE.
	0	Indicates not to modify the IP address.
	IP address	Specifies the new IP address to be used for this logical device image. When the device type (new or existing) is CP_LOCAL, this field contains the (new) IP address that is to be used for socket creation for the local logical device image. When the device type (new or existing) is CP_REMOTE, this field contains the (new) IP address that was used to create the socket interface for the CP_LOCAL logical device image. The *mupiipa* (existing or new) along with the new *mupiport* represent the address to be used to establish a socket connection path between this CP_REMOTE logical device image and the target CP_LOCAL logical device image.
mupiport		Specifies the port address for the local logical device image. This field is only valid if this request is issued for a new or existing device type of CP_REMOTE.
	0	Indicates not to modify the port address.
	port address	Specifies the (new) port address to be used for this logical device image. When the logical device image type (existing or new) is CP_REMOTE, this field contains the (new) port address for the local logical device image that was returned when the target CP_LOCAL logical device image was created. The *mupiipa* (existing or new) along with the *mupiport* (existing or new) represent the address to be used to establish a socket connection path between this CP_REMOTE logical device image and the target CP_LOCAL logical device image.

mupibuff		Indicates that when *mupitype* is CP_LOCAL, this field contains the minimum data buffer size that should be allocated (initially) for data transfer operations from or to the physical tape device driver.
	0	Indicates not to modify the data buffer size.
	length	Specifies the new data buffer size to be used for this logical device image.
usrspaceln		Specifies the length of the user space passed in the *usrspace* field. This field is only valid if this request is issued for a new or existing device type of CP_LOCAL.
	0	Indicates not to modify the length of the user space passed in the *usrspace* field.
	length	Specifies the new length of the user space passed in the *usrspace* field.
usrspace		Specifies the address in user space that was reserved by the caller of this request for use by the CP_LOCAL logical device image.
	0	Indicates not to modify the address in user space reserved for use by the CP_LOCAL logical device image.
	address	Specifies the new address in user space for use by the CP_LOCAL logical device for mapping “ill-behaved” **ioctl()** system calls. This address must remain for the duration of the existence of the CP_LOCAL logical device image. The size of the user space is passed in the *usrspaceln* field and needs to be equal to or larger than the maximum size **ioctl()** argument supported for the tape device represented by this CP_LOCAL logical device image.

Return Values

The fields contain the following values when this system call completes successfully.

On return, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_MUPIEVSN**	Specifies the block structure version number passed with the request.
secukey		Specifies the system security key passed with the request.
mupistatus		Indicates the request completion status for the modification of this logical device image. The defined entries include:
	REQ_COMP	Indicates that the request to modify this logical device image is complete. The logical device image is available for use.
	REQ_IN_PROCESS	Indicates that the request to modify this logical device image is in process, but not yet complete. The logical device image is not available for use at this time. Completion status for this logical device image modification will be reported through the normal SRSIE **ioctl()** system call mechanism (by TMPCP) for this pseudo tape device/logical device image.
devmmn		Specifies the *major/minor number* assigned to this device.
mupimm1		Contains the UPI *major/minor number* assigned by the device driver for this logical device image. This is always the value passed on entry.
mupimode		Indicates that the connection mode was established or changed for this logical device image. The defined entries include:
	MTE_NORMAL	Indicates that this logical device image is in MTE_NORMAL mode. TMPCP notification and UPI request queueing does not occur.

	MTE_NOTIFY	Indicates that this logical device image is in MTE_NOTIFY mode. When a UPI request is received for this logical device image, the request will be queued and notification will be provided through the normal SRSI **ioctl()** system call mechanism (by TMPCP) for this pseudo tape device driver.
mupiport	*address*	Specifies the port address for the newly-created socket interface if *mupitype* changed the logical device image to CP_LOCAL or if a new *mupiipa* value was provided for an existing CP_LOCAL device.
mupireqst		Specifies the state of a user (UPI) request for this logical device image. The defined entries include:
	NO_REQ	Indicates there is no user (UPI) system call outstanding for this logical device image.
	REQ_BLOCKED	Indicates that there is a user (UPI) system call outstanding for this logical device image. The system call is blocked within the logical device image waiting to present notification through the normal SRSIE **ioctl()** system call mechanism (by TMPCP) for this pseudo tape device/logical device image.
	REQ_NOTIFIED	Indicates that there is a user (UPI) system call outstanding for this logical device image. Notification through the SRSIE **ioctl()** system call mechanism was presented for a connection mode of MTE_NOTIFY. The system call is blocked within the logical device image waiting for one of the perform action (PACTION) subcommands of the SRSIE **ioctl()** system call to be issued to define the appropriate recovery action.
	REQ_PASSED	Indicates that there is a user (UPI) system call outstanding for this logical device image. The system call was passed to the physical tape device driver or to the CP_LOCAL logical device image (if this is a CP_REMOTE logical device image). **Note:** When this user request state is returned, the connection mode (*mupimode*) for this logical device image is implicitly set to MTE_NOTIFY unless the user level system call is blocked waiting for tape manager action.
curscst	*current call*	Specifies the current logical device system call being executed (or requested to be executed).
	CLOSE_ST	Indicates a close.
	IOCTL_ST	Indicates that an **ioctl()** was executed.
	NONE_ST	Indicates that a system call is not currently being executed.
	OPEN_ST	Indicates an open.
	READ_ST	Indicates a read.
	WRT_ST	Indicates a write.
	READV_ST	Indicates a read vector.
	WRTV_ST	Indicates a write vector.
next_state		Indicates Next Logical Image (or tape device driver) state information. This is used in conjunction with the *mupireqst* field to provide additional granularity relative to the state of a user (UPI) request for this logical device image, and to define the validity state of the *next_interrno* field. The defined entries include:
	NVALID_NPASS	Indicates the user (UPI) request for this logical device image (if any request is outstanding) was not passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image). Therefore, the *next_interrno* field does not contain valid information.

	NVALID_PASS	Indicates the user (UPI) request for this logical device image was passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image), but that device image has not returned completion status to the request. The *next_interrno* field does not contain valid information.
	VALID_RETU	Indicates the user (UPI) request for this logical device image was passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image) and that device image has returned completion status to the request. The *next_interrno* fields contain valid information.
next_interrno		Indicates that when the *next_state* field is VALID_RETU, this field contains the int value returned from the system call executed by the next logical image device (CP_LOCAL if this is a CP_REMOTE device, or the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image).
	0	Indicates that the system call request was successful.
	errno	Indicates that the system call request was not successful. This field contains the *errno* value returned.
next_rw_cnt		Indicates the read or write system call bytes transferred. When the *curscst* field is READ_ST, READV_ST, WRT_ST, or WRTV_ST, the *next_state* field is VALID_RETU, and the *next_interrno* field is 0 (success). This field contains the number of bytes transferred for a read or write system call request (as indicated by *curscst*) returned from the system call executed by the next logical image device (CP_LOCAL if this is a CP_REMOTE device, or the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image).
	nnnn	Indicates the number of bytes read or written as returned from the device driver.
next_ioctl		Specifies **ioctl()** system call attributes. When the *curscst* field is IOCTL_ST, this bit-significant field defines attributes of the **ioctl()** being executed.
	RESP_ARG	Passes back response information in the **ioctl()** argument (for example, Sense/Read Buffer Log).
	RESP_ARG_EIO	Passes back response information in the **ioctl()** argument when an **errno** of **EIO** is returned (for example, List Recording Format).
	STATE_CHG	Results in a state change (for example, Unload).
	SEQ_MV	Requests sequential movement (for example, Forward Space File).
	RAND_P_MV	Requests random predictable movement (for example, Locate Block ID, Rewind, and so on).
	NO_MV	Requests no movement.

A return value of 0 indicates a successful operation. A return value of -1 indicates the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EAGAIN] Indicates that the state of the logical device image prohibits the requested change at this time.

[EBUSY] Indicates that someone else is using the device specified in *fildes*.

[EINVAL] Indicates that the version field is out of date. Recompile the application.

[EINVAL] Indicates that either a security key mismatch was detected or that the security key was not established.

[EINVAL] Indicates that this **ioctl()** request was issued without the MPI bit set in the device minor number.

[EINVAL] Indicates an invalid *devmmn*, *mupimode*, *mupimm1*, or *mupitype* for the device.

[EINVAL] Indicates that the CP_REMOTE device received an invalid response from the CP_LOCAL logical device image.

[EINVAL] Indicates that an attempt was made to change a field that is not applicable to the logical device image type.

[ENOMEM] Indicates that the requested data buffer of size *mupibuff* was unavailable.

[ENXIO] Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.

[ ] Indicates that a socket interface could not be created for a CP_LOCAL or CP_REMOTE logical device image. The **errno** value from the function that failed is returned.

[ ] Indicates that a connection with the CP_LOCAL logical device image could not be established for a CP_REMOTE logical device image. The **errno** value from the function that failed is returned.

[ ] A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_rupi

Purpose

mte_rupi – Removes a user access path to a logical device image from the pseudo tape device driver configuration.

Syntax

#include <**sys**/**mte_tms.h**>

int ioctl(*fildes***,MTTMSRUPIE,***arg***);**
int *fildes*, **MTTMSRUPIE;**
struct rupie ****arg*;**

Parameters

fildes	Specifies the file descriptor returned from an **open()** system call.
MTTMSRUPIE	Specifies the **ioctl()** request to remove a user access path logical device entry from the MTE pseudo tape device driver configuration.
arg	Specifies the address of the structure passed.

Description

This **ioctl()** system call removes a logical device image created by a previous MTTMSCUPIE **ioctl()** system call. It can only be issued against a TMPCP device that was opened in Management Protocol Interface (MPI) mode.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to a device opened as a standard (non-DTMS) device.

The **rupie** structure is defined as follows:

```
struct rupie {
int     resvd;      /* Reserved                                  */
int     versn;      /* Version number field                      */
int     secukey;    /* Security key                              */
dev_t   upimm;      /* Device major/minor number                 */
int     subcmd;     /* Subcommand field                          */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_RUPIEVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established by the MTTMSCSKE **ioctl()**.
subcmd	**FORCE1**	Removes the logical device image if the logical device is not currently executing a system call.

	FORCE2	Removes the logical device image regardless of the state of the device. **Note:** Use of the FORCE1 or FORCE2 options can cause system integrity problems. It should only be used by someone who totally understands the environment and accepts responsibility for its action.
	PASSIVE	Removes the logical device image if the logical device is not open.
upimm	*major/minor number*	Specifies the *cupimm1 major/minor number* that was returned by the MTTMSCUPIE **ioctl()** system call when this logical device image was created. This entry is required.

Return Values

On successful completion, the UPI logical device image is removed.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY]	Indicates that someone else is using the device specified in *fildes*.
[EAGAIN]	Indicates that the logical device image being removed is still open and the *subcmd* field is not FORCE1 or FORCE2.
[EAGAIN]	Indicates that the logical device image being removed is currently executing a system call and the *subcmd* field is not FORCE2.
[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[EINVAL]	Indicates that either a security key mismatch was detected, or that the security key was not established.
[EINVAL]	Indicates that this **ioctl()** request was issued without the MPI bit set in the device minor number.
[EINVAL]	Indicates that an invalid device was specified in *upimm* (the device does not exist).
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing further device activity.
[ ]	A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_srsi

Purpose

mte_srsi – Sets and reads state information for a TMPCP.

Syntax

#include <**sys**/**mte_tms.h**>

int ioctl(*fildes***,MTTMSSRSIE,***arg***);**
int *fildes***,MTTMSSRSIE;**
struct tmssie ****arg*;**

Parameters

fildes	Specifies that the file descriptor returned from an **open()** system call.
MTTMSSRSIE	Specifies the **ioctl()** request to set or read state information for a tape management pseudo control point (TMPCP).
arg	Specifies the address of the **tmssie** structure passed.

Description

This **ioctl()** system call sets and reads state information for a TMPCP or a logical device image created by a previous MTTMSCUPIE **ioctl()** system call. The logical device image is referred to as the User Protocol Interface (UPI).

This **ioctl()** system call can only be issued against a TMPCP.

Note: On systems that support “C2” security, the user or process must have the “tmsmpi” privilege to open a device in MPI mode. This **ioctl()** system call cannot be issued to any device other than a TMPCP.

The **tmssie** structure is defined in as follows:

```
struct tmssie {
int      resvd;        /* Reserved                                    */
int      versn;        /* Version number field                        */
int      secukey;      /* Security key                                */
char     subcmd;       /* Subcommand field                            */
char     ddaction;     /* Device driver action code                   */
char     ddreason;     /* Device driver reason code                   */
char     upitype;      /* Logical device image type                   */
int      pwait;        /* Process wait (suspended) indicator          */
int      pwaitp;       /* Process wait pending notification indicator */
int      ddrcnt;       /* ddreason count                              */
int      tminterrno;   /* int/errno value to be returned to user      */
int      interrno;     /* int/errno value created by the pseudo DD    */
dev_t    upimm;        /* Major/minor number of the logical device    */
char     curscst;      /* Current system call state                   */
char     priscst;       /* Prior system call state                    */
u_short  timeOut;       /* Wait timeout value                         */
int      mtoflags;      /* O_Flags passed at open                     */
ulong    upireqst:      /* State of the user (UPI) request            */
int      next_state:    /* Next logical image called - state          */
int      next_interrno; /* Next logical image called - int/errno ret. */
int      next_rw_cnt;   /* Next logical image called - read/write cnt */
int      next_ioctl;    /* Next logical image called - ioctl type     */
};
```

On request, the fields contain:

resvd	**0**	Reserved.
versn	**TMS_SRSIEVSN**	Specifies the block structure version number.
secukey		Specifies the system security key. On systems that support the “tmsmpi” C2 Security privilege, this field is not used. On systems that do not support the “tmsmpi” C2 Security privilege, this field must contain the security key established with the MTTMSCSKE **ioctl()**.
subcmd	**PACTIONR**	Performs the action indicated in the *ddaction* field and returns when complete.
	PACTIONW	Performs the action indicated in the *ddaction* field and waits. Returns only when an error or exception condition is detected.
	PACTIONWT	Performs the action indicated in the *ddaction* field and waits. Returns when an error or exception condition is detected or following a timeout period specified in the timeOut field. The default is 10 minutes.
	RDSTATE	Returns the state information for this User Protocol Interface device.
	SRSIWAIT	Waits. Returns only when an error or exception condition is detected.
	WAITTIME	Waits. Returns when an error or exception condition is detected or following a timeout period specified in the timeOut field. The default is 10 minutes.
ddaction		Contains the action code to be performed by the device driver if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW.
	MT_CONTINUE	Continues the last operation.
	MT_RETRY	Retries the last operation. The actual retry or recovery procedure depends on the actual error reported.
	MT_RIERRNO	Returns the indication or **errno** value passed in the *tminterrno* field to the user.

	RETN_RW_CNT	Uses the value specified in the *next_rw_cnt* field to complete an outstanding user **write()** or **writev()** system call request, or **read()** or **readv()** system call request if the count value is 0. This *ddaction* request is accepted when issued to a CP_REMOTE logical device image with an outstanding **write()** or **writev()** system call request, if the value in the *next_rw_cnt* is greater than 0. It is also accepted for a **read()** or **readv()** system call if the value in the *next_rw_cnt* field is 0.
	RETN_IDATA	Uses the data and the length attribute provided by a previously issued **EUPI ioctl** that provided IDATA to this device image, to complete an outstanding user **read()** or **readv()** system call request. This *ddaction* request is only accepted when issued to a CP_REMOTE logical device image with an outstanding user **read()** or **readv()** system call request.
	RETN_IPARM	Uses the data and the length attribute provided by a previously issued **EUPI ioctl** that provided IPARM data to this device image, to complete an outstanding user **ioctl()** system call request. This *ddaction* request is only accepted when issued to a CP_REMOTE logical device image with an outstanding user **ioctl()** system call request.
tminterrno	**int/errno**	Contains the **int** or **errno** value to be returned to the user if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW and *ddaction* is RIERRNO.
upimm		Indicates that this field contains the *major/minor number* for the logical device image that is the subject of this command.
	0	Indicates that this request is for the TMPCP in general. When *subcmd* is SRSIWAIT or WAITTIME, this field is ignored. The WAIT is always executed on the TMPCP and any *ddreason* codes returned are related to the TMPCP or to a logical device created with this TMPCP. When *subcmd* is PACTIONR, PACTIONWT, or PACTIONW, this field must contain the *major/minor number* for the logical device image that the "Perform Action" (PACTION) portion of the command is addressed. This field is ignored for the WAIT portion of the PACTIONWT or PACTIONW *subcmd*.
	major/minor	Contains the *major/minor number* for the logical device image (associated with this TMPCP) that is the subject of this command if *subcmd* is PACTIONR, PACTIONWT, PACTIONW, or RDSTATE.
timeOut	*secs*	Specifies the number of seconds that this request should wait for an error or exception before returning with MT_TIMEOUT *ddreason* code.
	0	Specifies the timeout period. The default is 10 minutes.
next_rw_cnt	**nnnn**	Contains the **write** system call byte count value (or **read** if the value is 0) to be returned to the user if *subcmd* is PACTIONR, PACTIONWT, or PACTIONW, and *ddaction* is RETN_RW_CNT. The value being set cannot exceed the size of the current write operation.

Return Values

On successful completion, the *ddreason* field contains a reason code that reflects the condition that caused the device driver to return to the requestor. Except as noted in the individual reason code descriptions, the UPI process that set the *ddreason* field is suspended until a subsequent **ioctl(...MTTMSSRSIE...)** request is received that specifies a PACTION subcommand. The *pwait* field indicates whether the process that set the reason code is suspended waiting for tape management action to continue.

The fields contain the following values when this system call completes successfully:

resvd	**0**	Reserved.
versn	**TMS_SRSIEVSN**	Specifies the block structure version number.
ddreason	**CP_ERRNOINV**	Indicates that the tape manager returned an invalid *tminterrno*.
	CP_MISMAT	Indicates that a message-type failure was detected by a CP_LOCAL logical device image.
	CP_MSGCNTER	Indicates that a byte count mismatch was detected on data sent by a CP_REMOTE or CP_LOCAL logical device image.
	CP_RESPINV	Indicates that a message-type failure was detected by a CP_REMOTE logical device image.
	CP_SCF	Indicates that a socket connection failure was detected by a CP_LOCAL or CP_REMOTE logical device image.
	CP_SCF2	Indicates that a socket **accept()** failure was detected by a CP_LOCAL logical device image.
	CUPI_COMPLETE	Indicates that the previously issued Create User Protocol Interface **ioctl()** system call for this logical device image has completed. The status of that request is reflected in the **errno** value contained in the *interrno* field. The request that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process.
	IOCTL_SZ_ERR	Indicates that an **ioctl()** was issued against a UPI device and there is a mismatch between the CP_REMOTE node and the CP_LOCAL node on the expected argument size.
	MT_TIMEOUT	Indicates that the timeout period as specified by a PACTIONWT or WAITTIME *subcmd* expired.
	MUPI_COMPLETE	Indicates that a previously issued Modify User Protocol Interface **ioctl()** system call for this logical device image has completed. The status of that request is reflected in the **errno** value contained in the *interrno* field. The request that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process.
	NO_MEMORY	Indicates that a logical device image is unable to get memory.
	NO_WAIT	Indicates the normal completion of one of the NO_WAIT commands as defined in the *subcmd* field.
	NOTIFY_RTN	Indicates that an outstanding system call issued for this logical device has returned and one of the following occurred: • A previously issued Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call specified the *upimode* to be MTE_NOTIFY. • The MTE_NOTIFY mode for this logical device image was set implicitly by a previously issued Modify User Protocol Interface **ioctl()**.
	OPEN2	Indicates that another **open()** was attempted against the TMPCP. The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process.
	SVR_BRK_DOWN	Indicates that a CP_LOCAL socket-related failure occurred (see *interrno* for more information). The process that generated this reason code is not suspended and, therefore, no specific tape management action is required to resume that process.
	TOKEN_MISMAT	Indicates that the token passed in a client request message between a CP_REMOTE and a CP_LOCAL logical device image is invalid.

	UPIC	Indicates that the user issued a **close()** for this logical device and the previous Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call had specified the *upimode* as MTE_NOTIFY.
	UPII	Indicates that the user issued an **ioctl()** for this logical device and the previous Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call had specified the *upimode* as MTE_NOTIFY.
	UPIO	Indicates that the user issued an **open()** for this logical device and the previous Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call had specified the *upimode* as MTE_NOTIFY.
	UPIR	Indicates that the user issued a **read()** for this logical device and the previous Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call had specified the *upimode* as MTE_NOTIFY.
	UPITYPE_ERROR	Indicates that a UPI system call was issued against a logical device image type that does not accept system call requests (CP_LOCAL).
	UPIW	Indicates that the user issued a **write()** for this logical device and the previous Create User Protocol Interface or Modify User Protocol Interface **ioctl()** system call had specified the *upimode* as MTE_NOTIFY.
	VERSN_MISMAT	Indicates that the version number passed in a client request message between a CP_REMOTE and a CP_LOCAL logical device image pair does not match the version number in the receiving image.
upitype		Specifies the image type of the logical device that the state information represents.
	CP_LOCAL	Indicates that the logical device image is for a local connection point.
	CP_REMOTE	Indicates that the logical device image is for a remote connection point.
	MAGTAPE	Indicates that the logical device image is for a magnetic tape device.
	STUB	Indicates that the logical device image is for a null device.
pwait	*nnnn*	Indicates the wait status count. This field contains the number of processes that have presented a *ddreason* code that are suspended waiting for tape manager action (*ddaction*). When a *ddreason* code is presented from a logical device image, a zero value in this field indicates the process that presented this *ddreason* is not waiting, and a nonzero value indicates the process that presented this *ddreason* is waiting. When this field is returned representing TMPCP status (for example, a *ddreason* of MT_TIMEOUT or NO_WAIT is returned), this field contains the number of processes associated with this TMPCP, that have presented a *ddreason* code to the tape manager and are waiting for tape manager action.

pwaitp	*nnnn*	Specifies the wait status pending count. This field contains the number of processes that have posted a *ddreason* code (but have not yet presented that *ddreason* code) that are suspended waiting for tape manager action (*ddaction*). When a *ddreason* code is presented from a logical device image, a zero value in this field will indicate there are no processes waiting to present a *ddreason* to the tape manager, and a nonzero value will indicate the number of processes waiting (pending) to present a *ddreason* to the tape manager for this logical device image. When this field is returned representing TMPCP status (for example, a *ddreason* of MT_TIMEOUT or NO_WAIT is returned), this field will contain the number of processes associated with this TMPCP that have a pending *ddreason* code to be presented to the tape manager and are, therefore, waiting for tape manager action.
ddrcnt	*nnnn*	Specifies the number of *ddreason* codes waiting to be presented. When a *ddreason* code is presented from a logical device image, this count indicates the number of *ddreason* codes that this logical device image has remaining to present to the tape manager. When this field is returned representing TMPCP status (for example, a *ddreason* of MT_TIMEOUT or NO_WAIT is returned), it contains the number of *ddreason* codes waiting to be presented to the tape manager from logical device image processes associated with this TMPCP.
interrno	**int/error**	Indicates that if the *ddreason* field contains a reason code showing that an error or exception condition was detected by a logical device image, this field contains the **errno** value that best represents the condition detected. A value of zero indicates the device driver did not associate the *ddreason* code with a particular **errno**. If *ddreason* is SVR_RBK_DOWN, *interrno* will contain one of the following (see *ddreasons*) which further describes the exception: CP_SCF, CP_SCF2, CP_MSGCNTER, TOKEN_MISMAT, or CP_MISMAT.
upimm	*major/minor number*	Specifies the *major/minor number* of the logical device that the state information represents.
curscst	*current call*	Specifies the current logical device system call being executed (or requested to be executed).
	CLOSE_ST	Indicates a close.
	IOCTL_ST	Indicates that an **ioctl()** was executed.
	NONE_ST	Indicates that a system call is not currently being executed.
	OPEN_ST	Indicates an open.
	READ_ST	Indicates a read.
	WRT_ST	Indicates a write.
	READV_ST	Indicates a read vector.
	WRTV_ST	Indicates a write vector.
priscst	*prior call*	Specifies the prior system call executed by the logical device. Refer to the *curscst* field for a definition of the values that can be returned.
mtoflags	**O_flags**	Specifies the O_Flags passed by the last UPI **open()** system call. If the UPI device is not open or is not currently processing an **open()** system call, this field will contain -1.
upireqst		Specifies the state of a user (UPI) request for this logical device image. The defined entries include:

NO_REQ Indicates there are no user (UPI) system calls outstanding for this logical device image.

REQ_BLOCKED Indicates that there is a user (UPI) system call outstanding for this logical device image. The system call is blocked within the logical device image waiting to present notification through the normal SRSIE **ioctl()** system call mechanism, via the TMPCP for this pseudo tape device/logical device image.

REQ_NOTIFIED Indicates that there is a user (UPI) system call outstanding for this logical device image. Notification through the SRSIE **ioctl()** system call mechanism was presented for a connection mode of MTE_NOTIFY. The system call is blocked within the logical device image waiting for one of the perform action (PACTION) subcommands of the SRSIE **ioctl()** system call to be issued to define the appropriate recovery action.

REQ_PASSED Indicates that there is a user (UPI) system call outstanding for this logical device image. The system call was passed to the physical tape device driver or to the CP_LOCAL logical device image (if this is a CP_REMOTE logical device image).

Note: When this user request state is returned in response to a Modify User Protocol Interface **ioctl()**, the connection mode *mupimode* for this logical device image is implicitly set to MTE_NOTIFY unless the user-level system call is blocked waiting for tape manager action.

next_state Specifies the Next Logical Image (or tape device driver) state information. This is used in conjunction with the *upireqst* field to provide additional granularity relative to the state of a user (UPI) request for this logical device image, and to define the validity state of the *next_rw_cnt*, *next_ioctl*, and *next_interrno* fields. The defined entries include:

NVALID_NPASS Indicates the user (UPI) request for this logical device image (if any request is outstanding) was not passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image). Therefore, the *next_rw_cnt*, *next_ioctl*, and *next_interrno* fields do not contain valid information.

NVALID_PASS Indicates the user (UPI) request for this logical device image was passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the Tape Device Driver if this is a CP_LOCAL or MAGTAPE logical device image), but that device image has not returned completion status to the request. The *next_ioctl*, *next_rw_cnt*, and *next_interrno* fields do not contain valid information.

VALID_RETU Indicates the user (UPI) request for this logical device image was passed to the next logical image device (to CP_LOCAL if this is a CP_REMOTE device, or to the Tape Device Driver if this is a CP_LOCAL or MAGTAPE logical device image) and that device image has returned completion status to the request. The *next_ioctl*, *next_rw_cnt*, and *next_interrno* fields contain valid information.

next_interrno Indicates that when the *next_state* field is VALID_RETU, this field contains the int value returned from the system call executed by the next logical image device (CP_LOCAL if this is a CP_REMOTE device, or the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image). The values returned include:

0 Indicates that the system call request was successful.

	errno	Indicates that if a 0 was not returned, the system call request was not successful. This field contains the *errno* value returned.
next_rw_cnt		Indictes the read or write system call bytes transferred. When the *curscst* field is READ_ST or READV_ST or WRT_ST or WRTV_ST, the *next_state* field is VALID_RETU, and the *next_interrno* field is 0 (success). This field contains the number of bytes transferred for a **read** or **write** system call request (as indicated by *curscst*) returned from the system call executed by the next logical image device (CP_LOCAL if this is a CP_REMOTE device, or the tape device driver if this is a CP_LOCAL or MAGTAPE logical device image).
	nnnn	Indicates the number of bytes read or written as returned from the device driver.
next_ioctl		Specifies the **ioctl()** system call attributes. When the *curscst* field is IOCTL_ST and the *next_state* field is not NVALID_NPASS, this bit-significant field defines attributes of the **ioctl()** being executed.
	RESP_ARG	Passes back response information in the **ioctl()** argument (for example, Sense/Read Buffer Log).
	RESP_ARG_EIO	Passes back response information in the **ioctl()** argument when an **errno** of **EIO** is returned (for example, List Recording Format).
	STATE_CHG	Results in a state change (for example, Unload).
	SEQ_MV	Requests sequential movement (for example, Forward Space File).
	RAND_P_MV	Requests random predictable movement (for example, Locate Block ID, Rewind, and so on).
	NO_MV	Requests no movement.

A return value of 0 indicates a successful operation. A return value of -1 indicates that the operation did not complete successfully and **errno** is set to indicate the error.

Error Values

If the **ioctl()** fails, **errno** can be set to one of the following values:

[EBUSY]	Indicates that this request was issued to a TMPCP that has a previously issued request outstanding.
[EINTR]	Indicates that the request was interrupted by a signal.
[EINVAL]	Indicates that the version field is out of date. Recompile the application.
[EINVAL]	Indicates an invalid *subcmd* field value.
[EINVAL]	Indicates that the command was issued to a device other than a TMPCP.
[EINVAL]	Indicates that the subcommand is PACTIONR, PACTIONW, or PACTIONWT and the *ddaction* is not valid or inappropriate.
[EINVAL]	Indicates that the subcommand is PACTIONR, PACTIONW, PACTIONWT, or RDSTATE and the *upimm* field does not point to a valid logical device image.
[EINVAL]	Indicates that an attempt was made to send an action code to a UPI device that is not waiting for an action code.

[EINVAL]	Indicates that a security key mismatch was detected or that the security key was not established.
[EINVAL]	Indicates that this **ioctl()** request was issued without the MPI bit set in the device minor number.
[EINVAL]	Indicates an invalid *devmmn* field.
[ENXIO]	Indicates that an internal device driver error was detected. Close the device, unconfigure it, then reconfigure it before doing any further device activity.
[ ]	A **copyout()** or **copyin()** (kernel services) error occurred. Refer to the **copyout()** and **copyin()** commands for more information on the **errno** returned.

Related Information

Command: **mte_tmpcp**

mte_srsi

Chapter 4. SP Subroutines

© Copyright IBM Corp. 1996

getvhostname subroutine

Purpose

getvhostname – Gets the virtual host name.

Library

Availability Library (libavail.a)

Syntax

#include <**vhost.h**>
int getvhostname (*name*, *name_length*);
char **name*;
int *name_length*;

Parameters

name	Specifies the virtual host name.
name_length	Specifies the length of the *name* array.

Description

Use this subroutine to retrieve the virtual host name of a host machine. This routine is similar to the **gethostname** system call with the exception that it retrieves the virtual host name from the /**etc**/**vhostname** file instead of using a kernel variable. The **getvhostname** subroutine is a library call and **gethostname** is a system call.

The *name* is retrieved from the /**etc**/**vhostname** file. If the file does not exist, the **gethostname** system call is used and the real host name is returned. If excess space is provided, the returned *name* parameter is null terminated. If insufficient space is provided, the returned *name* parameter is truncated to fit in the given space. Virtual host names are limited to MAX_VHOSTNAME_LEN bytes (255), not including the terminating null character. The MAX_VHOSTNAME_LEN macro is defined in the **vhost.h** header file. In order to guarantee sufficient buffer space to hold the virtual host name, the *name_length* parameter should be MAX_VHOSTNAME_LEN + 1 or 256.

To clear the virtual host name so that the virtual host name no longer exists, remove the /**etc**/**vhostname** file.

Note: You must have root authority to remove the /**etc**/**vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may fail or act incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **getvhostname** instead of **gethostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the backup node. It should be noted that changing the real host

name in the kernel can cause problems with other applications that rely on the real host name to identify the *physical machine*.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set and clear the virtual host name in the supplied HACMP pre- and post-event scripts. The administrator should not normally have to set the virtual host name.

Return Values

Upon successful completion, the **getvhostname** subroutine returns a value of 0. Otherwise, a value of -1 is returned, the global variable **errno** is set to identify the error, and the contents of the buffer pointed to by the *name* parameter are indeterminate.

Error Values

The **getvhostname** subroutine is unsuccessful if the following error occurs:

EFAULT Indicates that either the *name* or *name_length* parameter gave an invalid address.

EINVAL Indicates that the *name_length* parameter is less than 0.

If one of the system calls used to retrieve the virtual host name from the /**etc**/**vhostname** file fails (for example, open or read), **errno** is set by the system call that failed.

Examples

1. To clear the virtual host name so that it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   Note: You must have root authority to remove the /**etc**/**vhostname** file.

2. To get the virtual host name from the /**etc**/**vhostname** file, enter:

   ```
   #include <vhost.h>
   main ( )
   {
   char name [MAX_VHOSTNAME_LEN + 1];
   getvhostname (name, (MAX_VHOSTNAME_LEN + 1));
   }
   ```

Related Information

Command: **vhostname**

Subroutine: **setvhostname**

AIX command: **hostname**

AIX subroutines: **gethostname**, **sethostname**

Header file: **vhost.h**

hacws_set subroutine

Purpose

hacws_set – Sets the state of the control workstation.

Library

Availability Library (libavail.a)

Location

/usr/lib/libavail.a

Syntax

#include <**hacws.h**>
int hacws_set (*state*);
int *state*;

Parameters

state Specifies the state of the control workstation. Valid values are: 0, 1, 2, 16, 32.

Description

Use this subroutine to set the current state of the control workstation. It is valid only when issued on the control workstation. When the subroutine is called and the calling process is not on a control workstation, an error occurs.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

Return Values

Upon successful completion, the **hacws_set** subroutine returns a value of 0. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

Error Values

The **hacws_set** subroutine is unsuccessful if any of the following errors occur:

EINVAL Indicates that the value of the *state* parameter is not one of the valid values contained in **hacws.h**.

ENODEV Indicates that the calling process is not on a control workstation.

ENOENT Indicates that the **hacws_set** subroutine could not determine whether the calling process is on a control workstation.

EPERM Indicates that the calling process did not have root's effective user ID.

If one of the system calls used to store the HACWS state value into the **/etc/hacws.state** file fails (for example, open, write, rename), **errno** is set by the system call that failed.

Macros

The **/usr/include/hacws.h** header file defines the following macros as valid input values for the **hacws_set** subroutine:

0 or **HACWS_NOT_AN_HACWS**
: Indicates that this control workstation is not in an HACWS configuration.

1 or **HACWS_PRIM_INACT_CWS**
: Indicates that this control workstation is the primary control workstation, but not the currently active control workstation.

2 or **HACWS_PRIM_ACT_CWS**
: Indicates that this control workstation is the primary control workstation and is the currently active control workstation.

16 or **HACWS_BACK_INACT_CWS**
: Indicates that this control workstation is the backup control workstation, but not the currently active control workstation.

32 or **HACWS_BACK_ACT_CWS**
: Indicates that this control workstation is the backup control workstation and is the currently active control workstation.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on the HACWS option.

Related Information

Commands: **lshacws**, **sethacws**

Subroutine: **hacws_set**

Header file: **hacws.h**

hacws_stat subroutine

Purpose

hacws_stat – Gets the state of the control workstation.

Library

Availability Library (libavail.a)

Location

/usr/lib/libavail.a

Syntax

#include <**hacws.h**>
int hacws_stat (*void*);

Description

Use this subroutine to return the current state of the control workstation. It returns an integer that indicates the state of the primary or backup control workstation and specifies whether the control workstation is a high availability configuration. This subroutine is valid only when issued on the control workstation. When the subroutine is called and the calling process is not on a control workstation, an error occurs.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

Return Values

Upon successful completion, the **hacws_stat** subroutine returns a nonnegative value. If the **hacws_stat** subroutine is unsuccessful, a value of -1 is returned and the global variable **errno** is set to identify the error.

Error Values

The **hacws_stat** subroutine is unsuccessful if any of the following errors occur:

ENODEV Indicates that the calling process is not on a control workstation.

ENOENT Indicates that the **hacws_stat** subroutine could not determine whether the calling process is on a control workstation.

ERANGE Indicates that the /**etc**/**hacws.state** file does not contain a valid HACWS state.

If one of the system calls used to retrieve the HACWS state value from the /**etc**/**hacws.state** file fails (for example, open or read), **errno** is set by the system call that failed.

Macros

The /**usr**/**include**/**hacws.h** header file defines the following macros for the nonnegative return values for the **hacws_stat** subroutine:

0 or **HACWS_NOT_AN_HACWS**
: Indicates that this control workstation is not in an HACWS configuration.

1 or **HACWS_PRIM_INACT_CWS**
: Indicates that this control workstation is the primary control workstation, but not the currently active control workstation.

2 or **HACWS_PRIM_ACT_CWS**
: Indicates that this control workstation is the primary control workstation and is the currently active control workstation.

16 or **HACWS_BACK_INACT_CWS**
: Indicates that this control workstation is the backup control workstation, but not the currently active control workstation.

32 or **HACWS_BACK_ACT_CWS**
: Indicates that this control workstation is the backup control workstation and is the currently active control workstation.

Prerequisite Information

Refer to *IBM Parallel System Support Programs for AIX: Administration Guide* for information on the HACWS option.

Related Information

Commands: **lshacws**, **sethacws**

Subroutine: **hacws_set**

Header file: **hacws.h**

setvhostname subroutine

Purpose

setvhostname – Sets the virtual host name.

Library

Availability Library (libavail.a)

Syntax

#include <**vhost.h**>

int setvhostname (*name*, *name_length*);
char **name*;
int *name_length*;

Parameters

name	Specifies the virtual host name.
name_length	Specifies the length of the *name* array.

Description

Use this subroutine to set the virtual host name of a host machine. Only programs with a root user ID can use this subroutine. This routine is similar to the **sethostname** system call with the exception that it stores the virtual host name in the /**etc**/**vhostname** file instead of using a kernel variable. The **setvhostname** subroutine is a library call and **sethostname** is a system call.

The *name* is stored in the /**etc**/**vhostname** file. If the file does not exist, it will be created. If it does exist, the file contents will be overwritten by the new virtual host name. Virtual host names are limited to MAX_VHOSTNAME_LEN bytes (255), not including the terminating null character. The MAX_VHOSTNAME_LEN macro is defined in the **vhost.h** header file. The *name_length* parameter does not have to allow for the terminating null character, therefore, the largest allowable value for *name_length* is MAX_VHOSTNAME_LEN.

To clear the virtual host name so that the virtual host name no longer exists, remove the /**etc**/**vhostname** file.

Note: You must have root authority to remove the /**etc**/**vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may fail or act incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **getvhostname** instead of **gethostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the backup node. It should be noted that changing the real host name in the kernel can cause problems with other applications that rely on the real host name to identify the *physical machine*.

Note: The High Availability Cluster Multiprocessing (HACMP) event scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set and clear the virtual host name in the HACMP pre- and post-event scripts. The administrator should not normally have to set the virtual host name.

Return Values

Upon successful completion, the **setvhostname** subroutine returns a value of 0. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

Error Values

The **setvhostname** subroutine is unsuccessful if the following error occurs:

EINVAL Indicates that the *name_length* parameter is greater than MAX_VHOSTNAME_LEN or less than 0.

EPERM Indicates that the calling process does not have an effective root user ID.

If one of the system calls used to store the virtual host name into the /**etc**/**vhostname** file fails (for example, open, write, rename), **errno** is set by the system call that failed.

Examples

1. To clear the virtual host name so that it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   Note: You must have root authority to remove the /**etc**/**vhostname** file.

2. To set the virtual host name to **spcw_prim**, enter:

   ```
   #include <string.h>
   #include <vhost.h>
   main ( )
   {
   char name[]="spcw_prim";
   setvhostname(name, strlen(name));
   }
   ```

Related Information

Command: **vhostname**

Subroutine: **getvhostname**

AIX command: **hostname**

AIX subroutines: **gethostname**, **sethostname**

setvhostname subroutine

Part 3. Appendix

© Copyright IBM Corp. 1996

Appendix A. Perspectives Colors and Fonts

Perspectives Colors with Red, Green, and Blue (RGB) Triplets

Note: Colors may vary depending on the type of display you are using.

The following list contains valid color names that can be supplied as optional arguments to the –**backgroundColor** and –**foregroundColor** flags:

Color	Red	Green	Blue
alice blue	240	248	255
antique white	250	235	215
aquamarine	127	255	212
azure	240	255	255
beige	245	245	220
bisque	255	228	196
black	0	0	0
blanched almond	255	235	205
blue	0	0	255
blue violet	138	43	226
brown	165	42	42
burlywood	222	184	135
cadet blue	95	158	160
chartreuse	127	255	0
chocolate	210	105	30
coral	255	127	80
cornflower blue	100	149	237
cornsilk	255	248	220
cyan	0	255	255
dark goldenrod	184	134	11
dark green	0	100	0
dark khaki	189	183	107
dark olive green	85	107	47
dark orange	255	140	0
dark orchid	153	50	204
dark salmon	233	150	122
dark sea green	143	188	143
dark slate blue	72	61	139
dark turquoise	0	206	209
dark violet	148	0	211
deep pink	255	20	147

© Copyright IBM Corp. 1996

Color	Red	Green	Blue
deep sky blue	0	191	255
dim gray	105	105	105
dodger blue	30	144	255
firebrick	178	34	34
floral white	255	250	240
forest green	34	139	34
ghost white	248	248	255
gold	255	215	0
goldenrod	218	165	32
gray	190	190	190
green	0	255	0
green yellow	173	255	47
honeydew	240	255	240
hot pink	255	105	180
indian red	205	92	92
ivory	255	255	240
khaki	240	230	140
lavender	230	230	250
lavender blush	255	240	245
lawn green	124	252	0
lemon chiffon	255	250	205
light blue	173	216	230
light coral	240	128	128
light cyan	224	255	255
light goldenrod	238	221	130
light gray	211	211	211
light pink	255	182	193
light salmon	255	160	122
light sea green	32	178	170
light sky blue	135	206	250
light slate blue	132	112	255
light slate gray	119	136	153
light steel blue	176	196	222
light yellow	255	255	224
lime green	50	205	50
linen	250	240	230
magenta	255	0	255
maroon	176	48	96
medium aquamarine	102	205	170
medium blue	0	0	205

Color	Red	Green	Blue
medium orchid	186	85	211
medium purple	147	112	219
medium sea green	60	179	113
medium slate blue	123	104	238
medium spring green	0	250	154
medium turquoise	72	209	204
medium violet red	199	21	133
midnight blue	25	25	112
mint cream	245	255	250
misty rose	255	228	225
moccasin	255	228	181
navajo white	255	222	173
navy blue	0	0	128
old lace	253	245	230
oldlace	253	245	230
olive drab	107	142	35
orange	255	165	0
orange red	255	69	0
orchid	218	112	214
pale goldenrod	238	232	170
pale green	152	251	152
pale turquoise	175	238	238
pale violet red	219	112	147
papaya whip	255	239	213
peach puff	255	218	185
peru	205	133	63
pink	255	192	203
plum	221	160	221
powder blue	176	224	230
purple	160	32	240
red	255	0	0
rosy brown	188	143	143
royal blue	65	105	225
saddle brown	139	69	19
salmon	250	128	114
sandy brown	244	164	96
sea green	46	139	87
seashell	255	245	238
sienna	160	82	45
sky blue	135	206	235

Color	Red	Green	Blue
slate blue	106	90	205
slate gray	112	128	144
snow	255	250	250
spring green	0	255	127
steel blue	70	130	180
tan	210	180	140
thistle	216	191	216
tomato	255	99	71
turquoise	64	224	208
violet	238	130	238
violet red	208	32	144
wheat	245	222	179
white	255	255	255
white smoke	245	245	245
yellow	255	255	0
yellow green	154	205	50

Perspectives Fonts

Note: Fonts will vary depending on the type of Xmachine or Xstation you are using.

The following list contains font names that can be supplied as optional arguments to the –**fontFamily** flag:

application
block
charter
clean
courier
ergonomic
fixed
helvetica
lucida
lucida bright
lucida typewriter
new century schoolbook
roman
sans serif
serif
special
terminal
times
times new roman
type
typewriter
utopia

Glossary of Terms and Abbreviations

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ANSI/EIA Standard - 440A: Fiber Optic Terminology*, copyright 1989 by the Electronics Industries Association (EIA). Copies can be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue N.W., Washington, D.C. 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

See. This refers the reader to multiple-word terms in which this term appears.

See also. This refers the reader to terms that have a related, but not synonymous, meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

This section contains some of the terms that are commonly used in the SP publications.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard *Vocabulary for Information Processing* (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3. ANSI definitions are preceded by an asterisk (*).

Other definitions in this glossary are taken from *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems* (SC20-1699) and *IBM DATABASE 2 Application Programming Guide for TSO Users* (SC26-4081).

© Copyright IBM Corp. 1996

A

address. A character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

AFS. A distributed file system that provides authentication services as part of its file system creation.

AIX. Abbreviation for Advanced Interactive Executive, IBM's licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications, including high function graphics and floating point computations.

AIX Version 3. The IBM AIX Version 3 for RISC System/6000 (also known as AIX/6000). It is based on UNIX System V, conforms with POSIX IEEE Standard 1003.1, is compatible with Berkeley Software Distribution 4.3 (4.3 BSD), and is designed to meet Trusted Computing Base level C2 security.

Amd. Berkeley Software Distribution automount daemon.

API. Application Programming Interface. A set of programming functions and routines that provide access between the Application layer of the OSI seven-layer model and applications that want to use the network. It is a software interface.

application. The use to which a data processing system is put; for example, a payroll application, an airline reservation application.

application data. The data that is produced using an application program.

ARP. Address Resolution Protocol.

ATM. Asynchronous Transfer Mode. (See *TURBOWAYS 100 ATM Adapter*.)

Authentication. The process of validating the identity of a user or server.

Authorization. The process of obtaining permission to perform specific actions.

B

batch processing. * (1) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation thus formed is processed or accomplished in the same run. * (2) The processing of data accumulating over a period of time. * (3) Loosely, the execution of computer programs serially. (4) Computer programs executed in the background.

BMCA. Block Multiplexer Channel Adapter. The block multiplexer channel connection allows the RS/6000 to communicate directly with a host System/370 or System/390; the host operating system views the system unit as a control unit.

C

call home function. The ability of a system to call the IBM support center and open a PMR to have a repair scheduled.

CDE. Common Desktop Environment. A graphical user interface for UNIX.

charge feature. An optional feature for either software or hardware for which there is a charge.

CLI. Command Line Interface.

client. * (1) A function that requests services from a server and makes them available to the user. * (2) A term used in an environment to identify a machine that uses the resources of the network.

Client Input/Output Sockets (CLIO/S). A software package that enables high-speed data and tape access between SP systems, AIX systems, and ES/9000 mainframes.

CLIO/S. Client Input/Output Sockets.

CMI. Centralized Management Interface provides a series of SMIT menus and dialogues used for defining and querying the SP system configuration.

connectionless. A communication process that takes place without first establishing a connection.

connectionless network. A network in which the sending logical node must have the address of the receiving logical node before information interchange can begin. The packet is routed through nodes in the network based on the destination address in the packet. The originating source does not receive an acknowledgment that the packet was received at the destination.

control workstation. A single point of control allowing the administrator or operator to monitor and manage the SP system using the IBM AIX Parallel System Support Programs.

css. Communication subsystem.

D

daemon. A process, not associated with a particular user, that performs system-wide functions such as administration and control of networks, execution of time-dependent activities, line printer spooling and so forth.

DASD. Direct Access Storage Device. Storage for input/output data.

DCE. Distributed Computing Environment.

DFS. distributed file system. A subset of the IBM Distributed Computing Environment.

DNS. Domain Name Service. A hierarchical name service which maps high level machine names to IP addresses.

E

Error Notification Object. An object in the SDR that is matched with an error log entry. When an error log entry occurs that matches the Notification Object, a user-specified action is taken.

ESCON. Enterprise Systems Connection. The ESCON channel connection allows the RS/6000 to communicate directly with a host System/390; the host operating system views the system unit as a control unit.

Ethernet. (1) Ethernet is the standard hardware for TCP/IP local area networks in the UNIX marketplace. It is a 10-megabit per second baseband type LAN that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by collision detection (CSMA/CD). (2) A passive coaxial cable whose interconnections contain devices or components, or both, that are all active. It uses CSMA/CD technology to provide a best-effort delivery system.

Ethernet network. A baseband LAN with a bus topology in which messages are broadcast on a coaxial cabling using the carrier sense multiple access/collision detection (CSMA/CD) transmission method.

event. In Event Management, the notification that a predicate evaluated to true. This evaluation occurs each time an instance of a resource variable is observed.

expect. Programmed dialogue with interactive programs.

F

failover. Also called fallover, the sequence of events when a primary or server machine fails and a secondary or backup machine assumes the primary workload. This is a disruptive failure with a short recovery time.

fall back. Also called fall back, the sequence of events when a primary or server machine takes back control of its workload from a secondary or backup machine.

FDDI. Fiber Distributed Data Interface.

Fiber Distributed Data Interface (FDDI). An American National Standards Institute (ANSI) standard for 100-megabit-per-second LAN using optical fiber cables. An FDDI local area network (LAN) can be up to 100 km (62 miles) and can include up to 500 system units. There can be up to 2 km (1.24 miles) between system units and/or concentrators.

File Transfer Protocol (FTP). The Internet protocol (and program) used to transfer files between hosts. It is an application layer protocol in TCP/IP that uses TELNET and TCP protocols to transfer bulk-data files between machines or hosts.

file. * A set of related records treated as a unit, for example, in stock control, a file could consist of a set of invoices.

file server. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

foreign host. Any host on the network other than the local host.

FTP. File transfer protocol.

G

gateway. An intelligent electronic device interconnecting dissimilar networks and providing protocol conversion for network compatibility. A gateway provides transparent access to dissimilar networks for nodes on either network. It operates at the session presentation and application layers.

H

HACMP/6000. AIX High Availability Cluster Multi-Processing/6000.

HACWS. High Availability Control Workstation function, based on HACMP/6000, provides for a backup control workstation for the SP system.

help key. In the SP graphical interface, the key that gives you access to the SP graphical interface help facility.

High Availability Cluster Multi-Processing/6000. An IBM facility to cluster nodes or components to provide high availability by eliminating single points of failure.

High Performance Switch. An IBM multi-stage packet switch for high-performance communication between processor nodes.

HiPPI. High Performance Parallel Interface. RS/6000 units can attach to a HiPPI network as defined by the ANSI specifications. The HiPPI channel supports burst rates of 100 Mbps over dual simplex cables; connections can be up to 25 km in length as defined by the standard and can be extended using third-party HiPPI switches and fiber optic extenders.

home directory. The directory associated with an individual user.

host. A computer connected to a network, and providing an access method to that network. A host provides end-user services.

HSD. The data striping device for the IBM Virtual Shared Disk. The device driver lets application programs stripe data across physical disks in multiple IBM Virtual Shared Disks, thus reducing I/O bottlenecks and hot spots.

I

instance vector. In Event Management, a set of elements, where each element is a name/value pair of the form `name=value`, whose values uniquely identify the copy of the resource (and by extension, the copy of the resource variable) in the system.

Internet. A specific inter-network consisting of large national backbone networks such as APARANET, MILNET, and NSFnet, and a myriad of regional and campus networks all over the world. The network uses the TCP/IP protocol suite.

Intermediate Switch Board. Switches mounted in the High Performance Switch expansion frame.

IP address. A 32-bit address assigned to devices or hosts in an IP internet that maps to a physical address. The IP address is composed of a network and host portion.

Internet Protocol (IP). (1) A protocol that routes data through a network or interconnected networks. IP acts as an interface between the higher logical layers and the physical network. This protocol, however, does not provide error recovery, flow control, or guarantee the reliability of the physical network. IP is a connectionless protocol. (2) A protocol used to route data from its source to it destination in an Internet environment.

ISB. Intermediate Switch Board.

K

Kerberos. A service for authenticating users in a network environment.

kernel. The core portion of the UNIX operating system which controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in “kernel mode” and is protected from user tampering by the hardware.

L

LAN. (1) Acronym for Local Area Network, a data network located on the user's premises in which serial transmission is used for direct data communication among data stations. (2) Physical network technology that transfers data a high speed over short distances. (3) A network in which a set of devices is connected to another for communication and that can be connected to a larger network.

local host. The computer to which a user's terminal is directly connected.

log database. A persistent storage location for the logged information.

log event. The recording of an event.

log event type. A particular kind of log event that has a hierarchy associated with it.

logging. The writing of information to persistent storage for subsequent analysis by humans or programs.

M

mask. To use a pattern of characters to control retention or elimination of portions of another pattern of characters.

menu. A display of a list of available functions for selection by the user.

Motif. The graphical user interface for OSF, incorporating the X Window System. Also called OSF/Motif.

MTBF. Mean time between failure. This is a measure of reliability.

MTTR. Mean time to repair. This is a measure of serviceability.

N

naive application. An application with no knowledge of a server that fails over to another server. Client to server retry methods are used to reconnect.

network. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

NFS. Network File System. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network, to access remote files in a LAN environment as though they were local files.

NIM. Network Installation Management is provided with AIX 4.1 to install AIX on the nodes.

NIM client. An AIX system installed and managed by a NIM master. NIM supports three types of clients:

- Standalone
- Diskless
- Dataless

NIM master. An AIX system that can install one or more NIM clients. An AIX system must be defined as a NIM master before defining any NIM clients on that system. A NIM master managers the configuration database containing the information for the NIM clients.

NIM object. A representation of information about the NIM environment. NIM stores this information as objects in the NIM database. The types of objects are:

- Network
- Machine
- Resource

NIS. Network Information System.

node. In a network, the point where one or more functional units interconnect transmission lines. A computer location defined in a network.

Node Switch Board. Switches mounted on frames that contain nodes.

NSB. Node Switch Board.

NTP. Network Time Protocol.

O

ODM. Object Data Manager. In AIX, a hierarchical object-oriented database for configuration data.

P

parallel environment. A system environment where message passing or SP resource manager services are used by they application.

Parallel Environment. A licensed IBM program used for message passing applications on the SP or RS/6000 platforms.

parallel processing. A multiprocessor architecture which allows processes to be allocated to tightly coupled multiple processors in a cooperative processing environment, allowing concurrent execution of tasks.

parameter. * (1) A variable that is given a constant value for a specified application and that may denote the application. * (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. * (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. * (4) A particular piece of information that a system or application program needs to process a request.

partition. See system partition.

Perl. Practical Extraction and Report Language.

pipe. A UNIX utility allowing the output of one command to be the input of another. Represented by the | symbol. It is also referred to as filtering output.

perspective. The primary window for each SP Perspectives application, so called because it provides a unique view of an SP system.

POE. Formerly Parallel Operating Environment, now Parallel Environment for AIX.

port. (1) An end point for communication between devices, generally referring to physical connection. (2) A 16-bit number identifying a particular TCP or UDP resource within a given TCP/IP node.

predicate. In Event Management, the relational expression between a resource variable and other elements (such as constants or the previous value of an instance of the variable) that, when true, generates an event. An example of a predicate is `X < 10` where X represents the resource variable `IBM.PSSP.aixos.PagSp.%totalfree` (the percentage of total free paging space). When the predicate is true, that is, when the total free paging space is observed to

be less than 10%, the Event Management subsystem generates an event to notify the appropriate application.

process. * (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. * (2) Any operation or combination of operations on data. * (3) A function being performed or waiting to be performed. * (4) A program in operation. For example, a daemon is a system process that is always running on the system.

protocol. A set of semantic and syntactic rules that defines the behavior of functional units in achieving communication.

PMR. Problem Management Report.

Problem Management Report. The number in the IBM support mechanism that represents a service incident with a customer.

Primary node or machine. (1) A device that runs a workload and has a standby device ready to assume the primary workload if that primary node fails or is taken out of service. (2) A node on the High Performance Switch that initializes, provides diagnosis and recovery services, and performs other operations to the switch network. (3) In IBM Virtual Shared Disk function, the node at which the logical volume is actually local and that acts as server node to I/O requests from other nodes.

R

RAID. Redundant array of independent disks.

rearm predicate. In Event Management, a predicate used to generate an event that alternates with an original event predicate in the following way: the event predicate is used until it is true, then the rearm predicate is used until it is true, then the event predicate is used, and so on. The rearm predicate is commonly the inverse of the event predicate (for example, a resource variable is on or off). It can also be used with the event predicate to define an upper and lower boundary for a condition of interest.

remote host. *See foreign host*.

resource. In Event Management, an entity in the system that provides a set of services. Examples of resources include hardware entities such as processors, disk drives, memory, and adapters, and software entities such as database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.

resource monitor. A program that supplies information about resources in the system. It can be a command, a daemon, or part of an application or subsystem that manages any type of system resource.

resource variable. In Event Management, the representation of an attribute of a resource. An example of a resource variable is `IBM.AIX.PagSp.%totalfree`, which represents the percentage of total free paging space. `IBM.AIX.PagSp` specifies the resource name and `%totalfree` specifies the resource attribute.

RISC. Reduced Instruction Set Computing (RISC), the technology for today's high performance personal computers and workstations, was invented in 1975. Uses a small simplified set of frequently used instructions for rapid execution.

rlogin (remote LOGIN). A service offered by Berkeley UNIX systems that allows authorized users of one machine to connect to other UNIX systems across a network and interact as if their terminals were connected directly. The rlogin software passes information about the user's environment (for example, terminal type) to the remote machine.

RPC. Acronym for Remote Procedure Call, a facility that a client uses to have a server execute a procedure call. This facility is composed of a library of procedures plus an XDR.

RSH. A variant of RLOGIN command that invokes a command interpreter on a remote UNIX machine and passes the command line arguments to the command interpreter, skipping the LOGIN step completely. See also *rlogin*.

S

SCSI. Small Computer System Interface.

server. (1) A function that provides services for users. A machine may run client and server processes at the same time. (2) A machine that provides resources to the network. It provides a network service, such as disk storage and file transfer, or a program that uses such a service. (3) A device, program, or code module on a network dedicated to providing a specific service to a network. (4) On a LAN, a data station that provides facilities to other data stations. Examples are file server, print server, and mail server.

Small Computer System Interface (SCSI). An input and output bus that provides a standard interface for the attachment of various direct access storage devices (DASD) and tape drives to the RS/6000.

shell. The shell is the primary user interface for the UNIX operating system. It serves as command language interpreter, programming language, and allows foreground and background processing. There

are three different implementations of the shell concept: Bourne, C and Korn.

Small Computer Systems Interface Adapter (SCSI Adapter). An adapter that supports the attachment of various direct-access storage devices (DASD) and tape drives to the RS/6000.

SMIT. The System Management Interface Toolkit is a set of menu driven utilities for AIX that provides functions such as transaction login, shell script creation, automatic updates of object database, and so forth.

SNMP. Simple Network Management Protocol. (1) An IP network management protocol that is used to monitor attached networks and routers. (2) A TCP/IP-based protocol for exchanging network management information and outlining the structure for communications among network devices.

socket. (1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing. (3) In TCP/IP, the Internet address of the host computer on which the application runs, and the port number it uses. A TCP/IP application is identified by its socket.

standby node or machine. A device that waits for a failure of a primary node in order to assume the identity of the primary node. The standby machine then runs the primary's workload until the primary is back in service.

subnet. Shortened form of subnetwork.

subnet mask. A bit template that identifies to the TCP/IP protocol code the bits of the host address that are to be used for routing for specific subnetworks.

subnetwork. Any group of nodes that have a set of common characteristics, such as the same network ID.

SUP. Software Update Protocol.

Sysctl. Secure System Command Execution Tool. An authenticated client/server system for running commands remotely and in parallel.

System Administrator. The user who is responsible for setting up, modifying, and maintaining the SP system.

subsystem. A software component that is not usually associated with a user command. It is usually a daemon process. A subsystem will perform work or provide services on behalf of a user request or operating system request.

syslog. A BSD logging system used to collect and manage other subsystem's logging data.

system partition. A group of nonoverlapping nodes on a switch chip boundary that act as a logical SP system.

T

tar. Tape ARchive, is a standard UNIX data archive utility for storing data on tape media.

Tcl. Tool Command Language.

TclX. Tool Command Language Extended.

TCP. Acronym for Transmission Control Protocol, a stream communication protocol that includes error recovery and flow control.

TCP/IP. Acronym for Transmission Control Protocol/Internet Protocol, a suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the underlying protocol is the Internet Protocol.

Telnet. Terminal Emulation Protocol, a TCP/IP application protocol that allows interactive access to foreign hosts.

Tk. Tcl-based Tool Kit for X Windows.

TMPCP. Tape Management Program Control Point.

token-ring. (1) Network technology that controls media access by passing a token (special packet or frame) between media-attached machines. (2) A network with a ring topology that passes tokens from one attaching device (node) to another. (3) The IBM Token-Ring LAN connection allows the RS/6000 system unit to participate in a LAN adhering to the IEEE 802.5 Token-Passing Ring standard or the ECMA standard 89 for Token-Ring, baseband LANs.

transaction. An exchange between the user and the system. Each activity the system performs for the user is considered a transaction.

transceiver (transmitter-receiver). A physical device that connects a host interface to a local area network, such as Ethernet. Ethernet transceivers contain electronics that apply signals to the cable and sense collisions.

transfer. To send data from one place and to receive the data at another place. Synonymous with move.

transmission. * The sending of data from one place for reception elsewhere.

TURBOWAYS 100 ATM Adapter. An IBM high-performance, high-function intelligent adapter that provides dedicated 100 Mbps ATM (asynchronous transfer mode) connection for high-performance servers and workstations.

U

UDP. User Datagram Protocol.

User Datagram Protocol (UDP). (1) In TCP/IP, a packet-level protocol built directly on the Internet Protocol layer. UDP is used for application-to-application programs between TCP/IP host systems. (2) A transport protocol in the Internet suite of protocols that provides unreliable, connectionless datagram service. (3) The Internet Protocol that enables an application programmer on one machine or process to send a datagram to an application program on another machine or process.

UNIX operating system. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers, but has been adapted for mainframes and microcomputers. **Note:** The AIX operating system is IBM's implementation of the UNIX operating system.

user. Anyone who requires the services of a computing system.

user ID. A nonnegative integer, contained in an object of type *uid_t*, that is used to uniquely identify a system user.

V

Virtual Shared Disk, IBM. The function that allows application programs executing at different nodes of a system partition to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

W

workstation. * (1) A configuration of input/output equipment at which an operator works. * (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

X

X Window System. A graphical user interface product.

Index

Special Characters

.klogin file 621

A

add_principal command 7
allnimres command 9
amq command 11
arp command 13

C

cfghsd command 15
cfghsdvsd command 16
cfgvsd command 17
chgcss command 19
cksumvsd command 21
cmonacct command 23
colors 737
commands
  add_principal 7
  allnimres 9
  amq 11
  arp 13
  cfghsd 15
  cfghsdvsd 16
  cfgvsd 17
  chgcss 19
  cksumvsd 21
  cmonacct 23
  cprdaily 24
  create_krb_files 25
  createhsd 27
  createvsd 32
  crunacct 38
  cshutdown 42
  CSS_test 48
  cstartup 49
  ctlhsd 54
  ctlvsd 56
  defhsd 59
  defvsd 61
  delnimclient 63
  delnimmast 65
  dsh 67
  dshbak 72
  Eannotator 74
  Eclock 77
  Eduration 82
  Efence 84
  emonctrl script 88
  Eprimary 94
commands *(continued)*
  Equiesce 97
  Estart 99
  Etopology 102
  Eunfence 104
  Eunpartition 106
  export_clients 107
  ext_srvtab 109
  ha.vsd 111
  hacws_verify 113
  haemcfg 115
  haemctrl script 119
  haemloadcfg 127
  haemtrcoff 130
  haemtrcon 133
  hagsctrl script 136
  hardmon 146
  hats 150
  hatsctrl script 152
  hb 158
  hbctrl script 161
  hc.vsd 166
  hmadm 168
  hmcmds 170
  hmmon 177
  hmreinit 198
  hostlist 200
  hr 202
  hrctrl script 204
  hsdatalst 208
  hsdvts 209
  ifconfig 210
  install_cw 216
  install_hacws 217
  jm_config 219
  jm_install_verify 221
  jm_start 223
  jm_status 225
  jm_stop 228
  jm_verify 230
  jmcmi_accesscontrol 232
  jmcmi_addpool 234
  jmcmi_changepool 237
  jmcmi_createjmdconfig 240
  jmcmi_deletepool 241
  jmcmi_servernodes 243
  kadmin 245
  kdb_destroy 250
  kdb_edit 251
  kdb_init 253
  kdb_util 254
  kdestroy 256

© Copyright IBM Corp. 1996

commands *(continued)*
Kerberos 623
kinit 259
klist 261
kpasswd 263
kprop 264
ksrvtgt 270
ksrvutil 271
kstash 273
locate_jm 274
lppdiff 275
lshacws 279
lshsd 281
lsvsd 283
mkamdent 286
mkconfig 287
mkinstall 289
mknimclient 290
mknimint 292
mknimmast 294
mknimres 296
ngaddto 298
ngclean 300
ngcreate 302
ngdelete 304
ngdelfrom 306
ngfind 308
nglist 310
ngnew 312
ngresolve 314
nodecond 316
nrunacct 318
p_cat 322
pcp 324
pdf 326
penotify 328
perspectives 330
pexec 333
pexscr 335
pfck 336
pfind 338
pfps 340
pls 343
pmanctrl 345
pmandef 349
pmanquery 356
pmanrmdloadSDR 358
pmv 360
ppred 362
pps 364
preparevsd 366
prm 368
psyslclr 370
psyslrpt 372
rcmdtgt 374
rcp 376

commands *(continued)*
removehsd 380
removevsd 382
resumevsd 384
rsh 386
s1term 569
SDR_test 389
SDRAddSyspar 391
SDRArchive 392
SDRChangeAttrValues 393
SDRClearLock 394
SDRCreateAttrs 395
SDRCreateClass 396
SDRCreateFile 397
SDRCreateObjects 398
SDRCreateSystemClass 399
SDRCreateSystemFile 400
SDRDeleteFile 401
SDRDeleteObjects 402
SDRGetObjects 403
SDRListClasses 405
SDRListFiles 406
SDRMoveObjects 407
SDRRemoveSyspar 408
SDRReplaceFile 409
SDRRestore 410
SDRRetrieveFile 412
SDRWhoHasLock 413
seqfile 414
services_config 416
sethacws 417
setup_authent 419
setup_CWS 422
setup_logd 424
setup_server 427
sp_configd 429
sp_configdctrl script 432
spacctnd 437
spacs_cntrl 439
spadaptrs 442
spapply_config 445
spbootins 448
spchuser 453
spcustomize_syspar 455
spcw_addevents 457
spcw_apps 458
spdeladap 460
spdelfram 462
spdelnode 464
spdisplay_config 466
spethernt 470
spevent 473
spframe 476
spget_syspar 478
sphardware 479
sphostnam 482

commands *(continued)*
sphrdwrad 484
splm 487
splst_syspars 494
splst_versions 495
splstdata 497
splsuser 501
spmkuser 502
spmon 504
spmon_ctest 509
spmon_itest 511
spperfmon 513
sprestore_config 516
sprmuser 518
spsitenv 519
spsvrmgr 523
spsyspar 530
spverify_config 533
spvsd 534
startvsd 537
statvsd 539
stopvsd 542
supper 545
suspendvsd 547
sysctl 549
sysctld 552
SYSMAN_test 557
syspar_ctrl 559
sysparaid 564
ucfghsd 571
ucfghsdvsd 572
ucfgvsd 574
unallnimres 576
undefhsd 578
undefvsd 579
updatehsd 580
updatevsdnode 582
updatevsdtab 585
verparvsd 587
vhostname 590
vsdatalst 592
vsddiag 595
vsdelnode 597
vsdelvg 599
vsdnode 601
vsdsklst 604
vsdvg 607
vsdvts 609
configuration, Event Management
haemloadlist file 614
cprdaily command 24
create_krb_files command 25
createhsd command 27
createvsd command 32
crunacct command 38
cshutdown command 42
CSS_test command 48
cstartup command 49
ctlhsd command 54
ctlvsd command 56

D

daemons
Emonitor 92
haemd 125
hagsd 142
hagsglsmd 144
kadmind 248
kerberos 257
kpropd 265
kshd 267
splogd 491
supfilesrv 544
defhsd command 59
defvsd command 61
delnimclient command 63
delnimmast command 65
dsh command 67
dshbak command 72
DTMS enablement function I/O control requests 649

E

Eannotator command 74
Eclock command 77
Eduration command 82
Efence command 84
emconditionctrl script 86
emonctrl script 88
Emonitor daemon 92
Emonitor subsystem
control script 88
introduction 88
Eprimary command 94
Equiesce command 97
Estart command 99
Etopology command 102
Eunfence command 104
Eunpartition command 106
Event Management
introduction 119
Event Management configuration
haemloadlist file 614
Event Management subsystem
control script 119
export_clients command 107
ext_srvtab command 109

F

files
.klogin 621
haemloadlist 614
hmacls 619
krb.conf 625
krb.realms 626
mte_exits 632
mte_itable 638
mte_make 639
SDR_dest_info 644
sysctl.acl 645
sysctl.conf 646
fonts 737

G

getvhostname subroutine 726
Group Services subsystem
control script 136

H

ha.vsd command 111
hacws_set subroutine 728
hacws_stat subroutine 730
hacws_verify command 113
haemcfg command 115
haemctrl script 119
haemd daemon 125
haemloadcfg command 127
haemloadlist file 614
haemtrcoff command 130
haemtrcon command 133
hagsctrl script 136
hagsd daemon 142
hagsglsmd daemon 144
hardmon command 146
hats command 150
hatsctrl script 152
hb command 158
hbctrl script 161
hc.vsd command 166
Heartbeat
introduction 161
Heartbeat subsystem
control script 161
hmacls file 619
hmadm command 168
hmcmds command 170
hmmon command 177
hmreinit command 198
Host_Responds
introduction 204
Host_Responds subsystem
control script 204
hostlist command 200
hr command 202
hrctrl script 204
hsdatalst command 208
hsdvts command 209

I

ifconfig command 210
install_cw command 216
install_hacws command 217
ioctl
mt_cassign 650
mt_cskey 653
mt_cupi 655
mt_gdci 658
mt_rdbd 664
mt_rupi 666
mt_srlf 668
mt_srmi 672
mt_srsi 678
mt_srsti 689
mt_tmcp 627
mte_cskey 694
mte_cupi 696
mte_eupi 701
mte_mupi 705
mte_rupi 712
mte_srsi 715
mte_tmpcp 641
IP source routing
setting required by Topology Services 150

J

jm_config command 219
jm_install_verify command 221
jm_start command 223
jm_status command 225
jm_stop command 228
jm_verify command 230
jmcmi_accesscontrol command 232
jmcmi_addpool command 234
jmcmi_changepool command 237
jmcmi_createjmdconfig command 240
jmcmi_deletepool command 241
jmcmi_servernodes command 243

K

kadmin command 245
kadmind daemon 248
kdb_destroy command 250

kdb_edit command 251
kdb_init command 253
kdb_util command 254
kdestroy command 256
Kerberos command 623
kerberos daemon 257
kinit command 259
klist command 261
kpasswd command 263
kprop command 264
kpropd daemon 265
krb.conf file 625
krb.realms file 626
kshd daemon 267
ksrvtgt command 270
ksrvutil command 271
kstash command 273

L

locate_jm command 274
lppdiff command 275
lshacws command 279
lshsd command 281
lsvsd command 283

M

manual pages for public code xxi
messaging
  required setting of IP source routing 150
mkamdent command 286
mkconfig command 287
mkinstall command 289
mknimclient command 290
mknimint command 292
mknimmast command 294
mknimres command 296
mt_cassign ioctl 650
mt_cskey ioctl 653
mt_cupi ioctl 655
mt_gdci ioctl 658
mt_rdbd ioctl 664
mt_rupi ioctl 666
mt_srlf ioctl 668
mt_srmi ioctl 672
mt_srsi ioctl 678
mt_srsti ioctl 689
mt_tmcp ioctl 627
mte_cskey ioctl 694
mte_cupi ioctl 696
mte_eupi ioctl 701
mte_exits file 632
mte_itable file 638
mte_make file 639
mte_mupi ioctl 705
mte_rupi ioctl 712
mte_srsi ioctl 715
mte_tmpcp ioctl 641

N

ngaddto command 298
ngclean command 300
ngcreate command 302
ngdelete command 304
ngdelfrom command 306
ngfind command 308
nglist command 310
ngnew command 312
ngresolve command 314
nodecond command 316
nonlocsrcroute option of no command
  setting required for Topology Services 150
nrunacct command 318

P

p_cat command 322
pcp command 324
pdf command 326
penotify command 328
perspectives command 330
pexec command 333
pexscr command 335
pfck command 336
pfind command 338
pfps command 340
pls command 343
pmanctrl command 345
pmandef command 349
pmanquery command 356
pmanrmdloadSDR command 358
pmv command 360
ppred command 362
pps command 364
preparevsd command 366
prm command 368
psyslclr command 370
psyslrpt command 372

R

rcmdtgt command 374
rcp command 376
removehsd command 380
removevsd command 382
restrictions
  Topology Services
    IP source routing setting 150

resumevsd command 384
routing, IP source
  setting required for Topology Services 150
rsh command 386

S

s1term command 569
scripts
  emconditionctrl 86
  emonctrl 88
  haemctrl 119
  hagsctrl 136
  hatsctrl 152
  hbctrl 161
  hrctrl 204
  sp_configdctrl 432
SDR_dest_info file 644
SDR_test command 389
SDRAddSyspar command 391
SDRArchive command 392
SDRChangeAttrValues command 393
SDRClearLock command 394
SDRCreateAttrs command 395
SDRCreateClass command 396
SDRCreateFile command 397
SDRCreateObjects command 398
SDRCreateSystemClass command 399
SDRCreateSystemFile command 400
SDRDeleteFile command 401
SDRDeleteObjects command 402
SDRGetObjects command 403
SDRListClasses command 405
SDRListFiles command 406
SDRMoveObjects command 407
SDRRemoveSyspar command 408
SDRReplaceFile command 409
SDRRestore command 410
SDRRetrieveFile command 412
SDRWhoHasLock command 413
seqfile command 414
services_config command 416
sethacws command 417
setup_authent command 419
setup_CWS command 422
setup_logd command 424
setup_server command 427
setvhostname subroutine 732
SP files 613
SP SNMP Proxy Agent subsystem
  control script 432
SP subroutines 725
SP technical information 613
sp_configd command 429
sp_configdctrl script 432
spacctnd command 437
spacs_cntrl command 439
spadaptrs command 442
spapply_config command 445
spbootins command 448
spchuser command 453
spcustomize_syspar command 455
spcw_addevents command 457
spcw_apps command 458
spdeladap command 460
spdelfram command 462
spdelnode command 464
spdisplay_config command 466
spethernt command 470
spevent command 473
spframe command 476
spget_syspar command 478
sphardware command 479
sphostnam command 482
sphrdwrad command 484
splm command 487
splogd daemon 491
splst_syspars command 494
splst_versions command 495
splstdata command 497
splsuser command 501
spmkuser command 502
spmon command 504
spmon_ctest command 509
spmon_itest command 511
spperfmon command 513
sprestore_config command 516
sprmuser command 518
spsitenv command 519
spsvrmgr command 523
spsyspar command 530
spverify_config command 533
spvsd command 534
startvsd command 537
statvsd command 539
stopvsd command 542
subroutines
  getvhostname 726
  hacws_set 728
  hacws_stat 730
  setvhostname 732
subsystem control scripts
  emonctrl 88
  haemctrl 119
  hagsctrl 136
  hatsctrl 152
  hbctrl 161
  hrctrl 204
  sp_configdctrl 432
supfilesrv daemon 544

supper command 545
suspendvsd command 547
sysctl command 549
sysctl.acl file 645
sysctl.conf file 646
sysctld command 552
SYSMAN_test command 557
syspar_ctrl command 559
sysparaid command 564

T

Topology Services
  introduction 152
  required setting of IP source routing 150
Topology Services subsystem
  control script 152
trademarks xi

U

ucfghsd command 571
ucfghsdvsd command 572
ucfgvsd command 574
unallnimres command 576
undefhsd command 578
undefvsd command 579
updatehsd command 580
updatevsdnode command 582
updatevsdtab command 585
utilities
  haemcfg 115
  haemloadcfg 127

V

verparvsd command 587
vhostname command 590
vsdatalst command 592
vsddiag command 595
vsdelnode command 597
vsdelvg command 599
vsdnode command 601
vsdsklst command 604
vsdvg command 607
vsdvts command 609